THE SOFTWARE HOUSE

Frontend Optimization Report

The Project Name

Date of report completion

12nd June 2023

Table of Contents

Introduction

! CHECK IF THE INTRODUCTION MATCHES THE BUSINESS'S NEEDS !

In the digital age, web applications are the **cornerstones** of many businesses' online presence. They act as the first point of contact for potential customers, a key driver for engagement and conversions, and as services at the heart of the business. The performance of a web application is not just a technical metric — it is a critical factor that influences user satisfaction, search engine rankings, and, ultimately, the success of a business.

This **Frontend Optimization Report** is designed to provide a comprehensive analysis and actionable recommendations to enhance the performance of your web application, ensuring it meets and exceeds the modern user's expectations. Performance optimization encompasses various aspects, including but not limited to reducing page load times, improving responsiveness, and ensuring smooth interactions.

This report will detail **your web application's** current performance metrics, identify bottlenecks and areas for improvement, and provide a tailored set of recommendations to enhance its frontend performance. We aim to equip you with the knowledge and action points needed to create a superior online experience that meets and exceeds your users' expectations.

By implementing the optimizations outlined in this report, you can ensure that your website stands out in a crowded digital landscape and **delivers real business results** through improved performance, user satisfaction, and engagement.

Reports summary

! WRITE THE SUMMARY DEPENDING ON REPORT RESULTS, EXAMPLE BELOW !

Our comprehensive review has identified several critical areas where performance is below industry standards, impacting user experience, conversion rates, and your digital presence's overall effectiveness.

Our analysis indicates that your site's average page load time significantly exceeds the ideal threshold of 2 to 3 seconds. This delay is primarily due to unoptimized images, excessive use of JavaScript, and lack of efficient caching strategies, leading to decreased user engagement and probably increased bounce rates. This inefficiency is particularly evident on mobile devices, where network conditions can further exacerbate loading times.

We have also discovered that the web application can freeze under certain conditions due to heavy computation being performed in the background. This makes the app unresponsive for a fraction of second, which worsens the user experience and makes your web application less

appealing to potential customers. Encountering such problems may lead users to expect that your business services could be similarly problematic.

Below, we listed all the issues identified during our analysis. These are categorized according to the priority of addressing them, based on their impact on user experience and conversion rates. In the later part of this report, each issue is described, and you can review the details of our findings, including information about their business consequences and strategies for resolution.

High priority

1.1	💥 Missing async and defer attributes
2.3	💥 Images are not using different size variants
2.4	💥 Some third-party scripts have overlapping functionality

Important

1.1	🔥 Missing async and defer attributes
2.3	🔥 Images are not using different size variants
2.4	🔥 Some third-party scripts have overlapping functionality

Suggestions

1.2	⚠️ Images are not correctly cached
2.5	⚠️ Images are not using different size variants
2.6	⚠️ Some third-party scripts have overlapping functionality

Implementation impact

Implementing the recommended fixes identified in our analysis addresses immediate performance issues and can significantly enhance your business outcomes. This section outlines the key benefits of implementing the proposed changes, illustrating how each contributes to building a stronger, more resilient, and profitable digital presence.

Better user experience

Optimizing the performance of your web application ensures that pages load quickly, which is crucial for retaining visitors. A faster app reduces the likelihood of users leaving (bounce rate) due to impatience, thereby increasing the chance of conversion. Moreover, a swift and smooth experience fosters customer satisfaction, encouraging repeat visits and positive word-of-mouth, which are invaluable for business growth.

> *GQ cut load time by 80% and saw an 80% increase in traffic. Median time spent on the site also increased by 32%. (source)*

Improved SEO results

Better performance has a real impact on how search engines will rate your website. Sites like Google or Bing favor fast web applications, and thus performance improvements can significantly increase the discoverability of web applications. Higher visibility in search results directly correlates with increased traffic, potentially attracting more customers and elevating your brand's online presence.

> *Rebuilding Pinterest pages for performance resulted in a 15% increase in SEO traffic and a 15% increase in conversion rate to signup. (source)*

Lower infrastructure costs

Optimizing content, compressing images, and minimizing the code on your website or application reduces the amount of data transferred between the server and the user. This decreased load means that servers can handle more concurrent users without compromising on performance, delaying the need for additional server capacity or more powerful hardware upgrades.

> *Netflix saw a 43% decrease in their bandwidth bill just after turning on GZip. (source)*

Enhanced product quality

Prioritizing performance and maintainability sets a high standard for quality within the development team. It encourages adopting best practices, thorough testing, and a proactive approach to code health. This culture leads to a more bug-free application and makes the application easier to maintain and extend over time, supporting business agility and innovation.

> *“The value of high-quality code can be difficult to communicate. Some managers see it as a boondoggle (...) since investing in code quality can slow development over the short term and doesn’t appear to alter the user experience. But nothing could be further from the truth.” (source)*

Limited environmental impact 🌿

Beyond the direct financial benefits, optimizing performance contributes to environmental sustainability. A more efficient use of server resources translates into lower energy consumption, which is not only good for the planet but also aligns with the growing consumer expectation for businesses to operate sustainably.

> *Shaving off a single kilobyte in a file that is being loaded on 2 million websites reduces CO2 emissions by an estimated 2950 kg per month. (source)*

Report details

1. Third-party scripts

💥 1.1 Missing async and defer attributes

Description	Script tags that include files from external servers should be loaded in a way that doesn't block executing the main application code. In the tested application, we found third-party scripts that are loaded in a synchronous way. Those script tags don't have the `async` or `defer` attributes. Loading non-essential third-party scripts in a synchronous manner may negatively impact the website by causing it to load needlessly slowly or become non-responsive for a period of time after loading
Business consequences	Longer loading times and application unresponsiveness make the user experience worse. The product starts to look less appealing, and the frustration grows. The likelihood of users abandoning the purchase funnel or onboarding process increases. *COOK, a frozen meals company, increased its conversion rate by 7% after cutting the average page load time by 0.85 seconds. The bounce rate also fell by 7%, and pages per session increased by 10%. (eggplantsoftware.com)*
Suggestions	Third-party script tags should have an async or a defer attribute, depending on a number of factors, like whether the script's order of execution is important or not. See: • https://web.dev/efficiently-load-third-party-javascript/ • https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

Technical details	• Tool: Google Chrome
	• Date: 28th March 2023
	• Address: https://example.com
	• Examples:
	  ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js
	  ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js

⚠️ 1.2 Missing DNS prefetch and preconnect resource hints

Description	When using third-party scripts, the browser can be informed to connect with the external server as soon as possible using resource hints dns-prefetch and preconnect, which helps with loading times of those resources. We found third-party scripts in the tested application but didn't find resource hints for one or more external servers.
Business consequences	Excessive loading times of analytics scripts can result in missed logging of some early user actions, which is bad for understanding customers' behavior. This means that ad scripts may be displayed very late, causing a late layout shift and thus worsening the user experience. This can lead to an increase in the drop rate. *iCook improved the cumulative layout shift by 15% and saw a 10% increase in ad revenue as a result. (web.dev)*
Suggestions	You should add resource hints `dns-prefetch` and `preconnect`, for every domain that your web application is downloading scripts from, for example, unpkg.com or googletagmanager.com. • https://www.smashingmagazine.com/2019/04/optimization-performance-resource-hints/ • https://developer.mozilla.org/en-US/docs/Web/Performance/dns-prefetch

Technical details	• Tool: Google Chrome
	• Date: 28th March 2023
	• Address: https://example.com
	• Examples:
	  ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js
	  ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js

🔥 1.3 Unnecessary third-party scripts are included

Description	Not all third-party scripts need to be included on page load. They should be loaded by the application only when they are needed to reduce the amount of data that the user is required to download. For instance, in a multipage app, we shouldn't include an external captcha script if the page doesn't have any forms that need protection against bots. In a single-page app, this script could be lazy loaded when a user navigates to a page with a form or even only after they scroll to a relevant page section (for example, when the form is not visible above the fold). In the application we tested, we found examples of scripts like that

	being loaded before they are really needed.
Business consequences	Including redundant scripts to early unnecessarily increases the overall loading time of the application, thereby degrading the user experience. This could detrimentally impact earnings without any justifiable reason. *Rossignol.com improved their load time by 1.9 seconds and cut their Speed Index by a factor of 10, contributing to a 94% improvement in conversion rate when compared to the year prior. (fasterize.com)*
Suggestions	Third-party scripts not in use on specific pages should be removed. It's important to determine which external scripts are required on every page, and all other files should only be included as needed. To take this a step further, the Intersection Observer API could be used to download third-party scripts just as users scroll to a given section of the page. • https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API • https://web.dev/intersectionobserver/
Technical details	• Tool: Google Chrome • Date: 28th March 2023 • Address: https://example.com/about.html • The following scripts are loaded too early:: ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js

⚠️ 1.4 Some third-party scripts have overlapping functionality

Description	Third-party scripts can enhance application functionalities, such as analytics, user behavior recording, or adding video players from popular services. To prevent the application from becoming too heavy, it should utilize only one service from each category.
Business consequences	Utilizing external scripts with overlapping functionalities can quickly cause the application to become bloated in size. This can lead to increased loading times, degrading the user experience, and thus raising the likelihood of users abandoning the purchase funnel or onboarding process. *SnipesUSA.com decreased page load speed by 30% and saw its average conversion rate double to from 1% to 2%. (digitalcommerce360.com)*
Suggestions	There should be a controlled list of external scripts used in the application and the product owner should monitor is there is no needless redundancy in the provided functionality, like for example, two logging services.

Technical details	• Tool: Google Chrome
	• Date: 28th March 2023
	• Address: https://example.com/about.html
	• Overlapping services:
	  ○ Live chats: Foo Chat, Bar Chat
	  ○ Ads: Google AdSense, Example Ads Service

⚠️ 1.5 Some scripts could be self-hosted

Description	Third-party scripts can sometimes be self-hosted and delivered from the same server as the application. This approach may improve download times and allow for better control over caching.
Business consequences	Self-hosting third-party scripts means your server has to send more data, but it facilitates improved caching. This can speed up the application's load time, enhancing user experience and potentially increasing conversions.
Suggestions	Consider revising the caching headers for some third-party scripts used by your application. If certain files are downloaded from the server too frequently, you might want to consider rehosting them on your own server with an improved caching configuration, either via a CDN or through modifications to the server config files.
Technical details	• Tool: Google Chrome • Date: 28th March 2023 • Address: https://example.com/about.html • Potential files: ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js

⚠️ 1.6 Use web workers to load third-party scripts

Description	Scripts can be loaded using web workers. This approach is beneficial for loading third-party scripts because then they don't impact the application's main thread. Web workers operate in a separate thread. As a result, the application's rendering speed is improved.
Business consequences	Third-party scripts are often non-essential, but loading, parsing, and executing them may render the app unresponsive to the user during the initial few seconds. This can degrade the user experience, which may in turn harm the application's revenue.
Suggestions	Consider using the Partytown library to shift the loading of third-party

	scripts from the main thread to web workers.
Technical details	• Tool: Google Chrome • Date: 28th March 2023 • Address: https://example.com/about.html • Examples: ○ https://example.net/production/assets/something/very/long/url/app.8Dt2ubp26j.js

2. Images

🔥 2.1 Image are using old file formats (jpg, png, gif)

Description	During the investigation we've found many images that are using old-gen file formats like JPG, PNG. There is a room for improvements, by converting them to WebP, reducing their size significantly. *WebP lossless images are 26% smaller compared to PNGs. WebP lossy images are 25-34% smaller than comparable JPEG images at equivalent SSIM quality index.* *(https://developers.google.com/speed/webp)*
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Images should use next-gen format like WebP or JPEG-2000, which are significantly lighter than traditional PNG or JPG. We can use either online converts for it, or convert it in code using NPM packages. • https://cloudconvert.com/webp-converter • https://www.npmjs.com/package/sharp
Technical details	• Tool: WebPageTest • Date: 21st May 2023 • Address: https://example.com • Examples: https://example.net/production/assets/something

💥 2.2 Images are not using different size variants

Description	Application is not using smaller image variants on mobile and tablet devices. Because of that, the website is fetching full size images on smaller viewport, which leads to overload of resources.

Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Each image should have multiple size variants generated, to be served on different breakpoints. They should be served either by using **<picture>** tag or **<img>** tag with **imgSrc** and **sizes** attributes. The browser will be responsible for loading the proper variant on each viewport size. To generate multiple variants out of a single source image, use: • https://www.npmjs.com/package/sharp
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 2.3 Images outside the viewport are not lazy loaded

Description	During the initial load of the application, many images which are placed outside the initial viewport were loaded. This leads to more resources download at the very beginning of the load process, which slows down the app.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Images that are Below The Fold (**BTF**) or outside the initial viewport, should be lazy loaded, to limit the amount of resources downloads during the initial app load. To achieve that, use native browser lazy loading by adding **lazy** attribute to those **<img>** tags • https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 2.4 LCP Image is not preloaded

Description	Image recognized as Largest Contentful Paint (**LCP**) was not preloaded, which means that the browser will only start fetching this image, when it encounters it during the parsing process. Because of that, many other less crucial resources would be downloaded before, which would impact the LCP metric.
Business consequences	LCP is one of the crucial Core Web Vitals metric. It impacts 25% of the whole Performance metric score. The longer the metric is, the worse is the Core Web Vitals score is, which can impact UX of the app, as well as revenue and SEO.
Suggestions	An image that is considered LCP by Lighthouse should be preloaded, to speed up it's download time and priority. To achieve that, use native browser **<link>** tag with **rel="preload"** attribute • https://web.dev/preload-responsive-images/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 2.5 Images have not defined size or aspect ratio

Description	Images does not have either **width** and **height** or **aspect-ratio** attributes defined. This might lead to layout shifts.
Business consequences	Images without statically defined **width** and **height** or **aspect-ratio** attributes might cause shifts in layout, otherwise known as Cumulative Layout Shift (**CLS**). This metric impacts both UX leading to unexpected user behavior, and the Lighthouse Performance score.
Suggestions	Each image should have **width** and **height** statically defined if it's known, or at least **aspect-ratio** attribute. In this way, browser would preserve needed space for image to load, without any unexpected CLS. • https://developer.mozilla.org/en-US/docs/Web/CSS/aspect-ratio • https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com

- Examples:https://example.net/production/assets/something

🔥 2.6 Images are not properly cached

Description	Images on the website does not have proper caching policy set up. This leads to fetching the resources multiple times during multiple visits to the application.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Images should be cached for as long as possible. There are many CDN providers out there: • https://cloudinary.com/ • https://www.cloudflare.com/cdn/ • https://www.twicpics.com/product/image
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

3. Fonts

💥 3.1 Font files are not using optimal extension

Description	Font files are not using **.woff2** extension, which usually is much lighter than **.woff, .ttf or .eot** extensions.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	Replace the font files with **.woff2** extension. Either find a proper files on the internet or use font converters. • https://www.fontconverter.io/en • https://web.dev/reduce-webfont-size/

Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 3.2 Font is not using subsets

Description	The font, app is using offers subsets, which are smaller version of a given font with limited number of glyphs. For example latin subset will only contain latin letters and characters.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	Check whether the font you are using offers subsets suitable for the app needs, and replace the source files with them. • https://fonts.google.com/knowledge/glossary/subsetting
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 3.3 Font file is not preloaded

Description	Font files are not preloaded, which extends its download time.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	If the font-face declaration is not inlined then the preload is a must-have. In other cases, it's nice to have. Add the **preload** resource hint to font links. • https://web.dev/codelab-preload-web-fonts/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 3.4 Font-face have invalid font-display attribute value

Description	Font-face declaration is not using **swap** value of **font-display** attribute. **Swap** value has extremely small block period and an infinite swap period which means that fallback font will be displayed until the dedicated font loads. This eliminates **FOUT** and **CLS**
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	Change the **font-display** value to **swap** • https://developer.chrome.com/blog/font-display/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 3.5 Font-face declaration is not inlined

Description	Font-face declaration is not optimal and slower.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	The optimal way of declaring **font-face** is inlining them in **<style>** tag in the **<head>** section of the DOM. This makes it Critical CSS. • https://developer.mozilla.org/en-US/docs/Web/CSS/@font-face#specifying_a_downloadable_font • https://www.smashingmagazine.com/2015/08/understanding-critical-css/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 3.6 Font is not self-hosted

Description	Font files are hosted on a 3rd party server. This adds additional external request (DNS-lookup, SSL Negotiation, TCP handshake) which adds time to download.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	Font files should be downloaded from the same origin as the application, to remove the need for external requests. • https://web.dev/patterns/web-vitals-patterns/fonts/font-self-hosted/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

4. Chunks & Bundles

🔥 4.1 App is not using chunk-splitting and code-splitting

Description	App bundles are big, and it takes a long time to download them.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	App chunks should be separated into smaller ones. This can be achieved by either chunks-splitting or code-splitting • https://webpack.js.org/plugins/split-chunks-plugin/#optimizationsplitchunks • https://webpack.js.org/guides/code-splitting/
Technical details	• Tool: bundle-wizard • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 4.2 Duplicated resources in chunks

Description	A couple of libraries like X or Y are duplicated across multiple chunks. This leads to redundant resources downloads, which only extends the app load time.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Libraries X and Y should be separated into their own chunk, and this should be the only place where its code is loaded. To achieve that, we can configure Webpack to merge chunks into single bundle • https://webpack.js.org/configuration/optimization/#optimizationmergeduplicatechunks
Technical details	• Tool: bundle-wizard • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 4.3 Lots of unused code in core chunks

Description	In core chunk X, there is X% of unused code, which extends the size of it unnecessarily. Core chunks with most crucial functionalities and code should load as soon as possible, and should be as lean as possible.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Code-split the part of the chunk that is not being used by it. See: • https://webpack.js.org/guides/code-splitting/
Technical details	• Tool: bundle-wizard • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

5. Styles

💥 5.1 Style files are loaded inefficiently

Description	CSS files are render blocking resources, so it's crucial to load them efficiently, to not extend the load time of the application.
Business consequences	Slow CSS loading can lead to bad UX, where Flash Of Unstyled Content (**FOUC**) can appear. Additionally, core Performance metrics like First Contentful Paint (**FCP**) can suffer.
Suggestions	CSS files should be loaded via **<link>** tags in the **<head>** section of the DOM. If you have multiple files with CSS, you should aim to **@import** them in the root CSS file. See: • https://developer.mozilla.org/en-US/docs/Web/CSS/@import • https://www.quickonlinetips.com/archives/2013/05/css-files-load-faster/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 5.2 Styles are not separated per viewport dimensions

Description	Styles specific for given breakpoints can be separated to different files, which could later on be fetched only on matching viewport size. This would result in decreasing the amount of CSS loaded in the app, which would be faster.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Consider splitting CSS into different files per breakpoints. You don't have to do it manually there are packages out there, which automates this process: • https://github.com/SassNinja/media-query-plugin • https://github.com/SassNinja/postcss-extract-media-query Later on, you can use **@import** statement with media queries condition to load:

⚠️ 5.2 Styles are not separated per viewport dimensions

Description	Styles specific for given breakpoints can be separated to different files, which could later on be fetched only on matching viewport size. This would result in decreasing the amount of CSS loaded in the app, which would be faster.
	• https://developer.mozilla.org/en-US/docs/Web/CSS/@import
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 5.3 Crucial styles are not considered Critical CSS

Description	Styles, necessary for rendering initial viewport, should be considered as Critical CSS to be loaded as soon as possible. Inlining styles in **<style>** tag in **<head>** section of the DOM, eliminates the need for external requests and speeds up the rendering process.
Business consequences	Slow CSS loading can lead to bad UX, where Flash Of Unstyled Content (**FOUC**) can appear. Additionally, core Performance metrics like First Contentful Paint (**FCP**) can suffer.
Suggestions	Styles needed for content Above The Fold (**ATF**) should be extracted to a separate inlined declaration in **<style>** tag, under **<head>** section of the DOM. See: • https://web.dev/extract-critical-css/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 5.4 App does not utilize content-visibility property

Description	The **content-visibility** property, enables the user agent to skip an element's rendering work, including layout and painting, until it is needed.

Business consequences	The more resources our application have to render, the longer the load time will be.
Suggestions	For the elements that are Below The Fold (**BTF**) use the **content-visibility** property, to limit renders during the initial load See: • https://developer.mozilla.org/en-US/docs/Web/CSS/content-visibility • https://web.dev/content-visibility/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

💥 5.5 Styles contains lots of unused code

Description	Much of the CSS code which is fetched is not used on given page. This leads to heavier files, and more resources usage.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Try to eliminate as much of unused CSS as possible. Chrome DevTools can help you with identifying percentage of usage of specific files. Or you can try automating this by using npm packages like **purgecss**. See: • https://developer.chrome.com/docs/lighthouse/overview/ • https://purgecss.com/
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 5.6 App uses expensive animation properties

Description	Some of the animations used in the app are causing Layout and Paint phase recalculations which are impacting the website Performance.

Business consequences	Animations which are not smooth might cause bad UX and users frustrations.
Suggestions	Adjust current animations so that they use rather **opacity, filter** or **transform** properties, which are not causing any side effects in the rendering process. Additionally, try to use **will-change** property, for further performance gains. See: • https://web.dev/animations-guide/ • https://developer.mozilla.org/en-US/docs/Web/Performance/CSS_JavaScript_animation_performance • https://developer.mozilla.org/en-US/docs/Web/CSS/will-change
Technical details	• Tool: Lighthouse • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

6. Google Tag Manager (only if used)

🔥 6.1 Scripts are injected too early

Description	Google Tag Manager offers lots of different trigger types (events on which the tag would be executed). Choosing a proper one have a great impact on Performance.
Business consequences	Scripts loaded via Google Tag Manager shouldn't interfere with website Performance. The earlier the script is loaded, the higher chance is that it will block other crucial resources from executing, which might cause in a longer wait time for the user and drop in conversion.
Suggestions	The rule of thumb for scripts loaded via Google Tag Manager is to load them as late as possible. Majority of the scripts should use **Window Loaded** trigger type, so they'll be injected after the performance is already measured. See: • https://support.google.com/tagmanager/answer/7679319?hl=en • https://web.dev/tag-best-practices/
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023

- Address: https://example.com
- Examples:https://example.net/production/assets/something

🔥 6.2 Scripts are injected into <head> section

Description	Lots of 3rd party providers want their script to be injected as soon as possible. As a result, they often provide a code snippet for script injection, where their script is injected as the first one, on the top of the <head> section. Usually, by the time Google Tag Manager container was executed on the website, the <head> section of the DOM was already parsed. Injecting anything into it after the initial parse, would force the browser to re-calculate and reparse it once again, which takes time.
Business consequences	All the DOM calculations like painting, rendering, parsing takes time. Extending it, would slow down the website, and influence the performance score.
Suggestions	Avoid injecting scripts into the **<head>** section. Instead, inject them at the end of the **<body>** section, where it would not cause any extra work for the browser. See: • https://web.dev/tag-best-practices/
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 6.3 Website is using multiple GTM containers

Description	Using multiple containers on a single page can create significant performance issues as it introduces additional overhead and script execution. At the very least, it duplicates the core tag code itself which, as it is delivered as part of the container's JavaScript, cannot be reused between the containers.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Try to eliminate extra containers, and load only one. See:

	• https://web.dev/tag-best-practices/
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 6.4 GTM container is heavy

Description	The size of a container is determined by its tags, triggers, and variables. Although a small container may still negatively impact page performance, a large container almost certainly will.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Try to keep the container size as small as possible. Remove duplicated and not used tags, triggers and variables. The recommended size is below **140kb.** See: • https://web.dev/tag-best-practices/
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 6.5 GTM injects HTML into the code

Description	Google Tag Manager allows us to inject custom HTML code into DOM of the application. This might be dangerous, if the content gets injected Above The Fold (**ATF)**, because it could generate a Cumulative Layout Shift (**CLS**).
Business consequences	Cumulative Layout Shift (**CLS**) is one of the key Lighthouse Performance metrics. Keeping it low increases the overall score of our application. Additionally, it has a great impact on the User Experience (**UX**), and might lead to user frustration or unintentional behavior.
Suggestions	Avoid injecting HTML code into the app through GTM tags.

	See: • https://web.dev/tag-best-practices/
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

7. React (only if used)

⚠️ 7.1 Components get re-rendered many times

Description	Re-renders in React happens whenever there is a change in the component and the UI needs to be updated.
Business consequences	Renders in React are fast, but not immediate. The more times the component gets re-rendered, the more time it would take. This might cause lower Performance metrics as well as worse UX of the application
Suggestions	Investigate core components which are re-rendered the most using tools like: • https://react.dev/learn/react-developer-tools • https://github.com/welldone-software/why-did-you-render And try to eliminate as much of them as possible.
Technical details	• Tool: React DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 7.2 Components are not code-split

Description	React allows us to code-split the code into separate JS chunks. And load them on demand, whenever user needs them. This reduces the amount of resources that have to be downloaded during the initial load of our app.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more

of their mobile data resources to access our website.

Suggestions	Try to code-split components that are only showed on user interaction (onClick, onScroll etc..). Components like Modals, Dropdowns or Accordions are perfect candidates to be split into separate chunks. See: • https://legacy.reactjs.org/docs/code-splitting.html
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

⚠️ 7.3 Application is not utilizing memoization methods

Description	React provides a couple of memoization methods which should reduce the amount of re-renders and should improve speed of given method execution.
Business consequences	Renders in React are fast, but not immediate. The more times the component gets re-rendered, the more time it would take. This might cause lower Performance metrics as well as worse UX of the application
Suggestions	Utilize React memoize methods to improve performance, and reduce the amount of rerenders. See: • https://react.dev/reference/react/useMemo • https://react.dev/reference/react/useCallback • https://react.dev/reference/react/memo
Technical details	• Tool: Chrome DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

8. Next.js (only if used)

💥 8.1 App is not using Next/Image component

Description	Next.js shares a custom image component, which handles various performance optimizations like lazy loading, preloading, conversion to .webp and generating different size variants.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Use dedicated Next/Image component for all the images in the app if not using any other form of optimizations. See: • https://nextjs.org/docs/pages/api-reference/components/image
Technical details	• Tool: React DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 8.2 App is not using Next/Script component

Description	Next.js shares a custom script component, which handles various performance optimizations like lazy loading, preloading and different loading strategies.
Business consequences	The more resources our application have to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Use dedicated Next/Script component for all the scripts in the app if not using any other form of optimizations. See: • https://nextjs.org/docs/pages/api-reference/components/script
Technical details	• Tool: React DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

🔥 8.3 App is not using Next/Font component

Description	Next.js shares a custom font component, which handles various performance optimizations like preloading, subsets, font-display and more.
Business consequences	The longer it takes to load a dedicated font, the bigger possibility there is for Flash Of Unstyled Text (**FOUT**), which leads to bad UX and possibly even a CLS.
Suggestions	Use dedicated Next/Font component for all the fonts in the app if not using any other form of optimizations. See: • https://nextjs.org/docs/pages/api-reference/components/font
Technical details	• Tool: React DevTools • Date: 21st May 2023 • Address: https://example.com • Examples:https://example.net/production/assets/something

9. JavaScript

🔥 9.1 App is not using modern browser API's

Description	Modern browsers API's are offering a lot of functionalities which helps us to resolve a lot of common issues. Using these functionalities instead of external dependencies can lead to smaller bundle size.
Business consequences	The more resources our application has to download, the longer the load time will be. Additionally, our clients would have to spend more of their mobile data resources to access our website.
Suggestions	Use a browser API instead of these dependencies: • lodash.filter • ...
Technical details	• In foo.ts you are using lodash.filter, use Array.filter instead • Date: 19th September 2023 • File path: src/users/utils/foo.ts

🔥 9.2 App is not using throttle/debounce

Description	Reducing the number of code calls may have a positive impact on the CPU and the number of requests.
Business consequences	The more CPU applications are using and the more requests it's doing there's a higher possibility for bad UX due to performance.
Suggestions	Use throttle/debounce
Technical details	• There's a code running on each scroll event, application's performance is suffering because of missing throttle/debounce on scroll event • Date: 19th September 2023 • File path: src/components/table/utils.ts

🔥 9.3 App is running heavy computations

Description	Heavy computations may perform very poorly on low-end devices.
Business consequences	The more CPU applications are using there's a higher possibility for bad UX due to performance.
Suggestions	Look for algorithm improvements for reducing the number of operations/memory required. Use console.time to measure time needed to execute given part of code
Technical details	• You're having an algorithm that does a lot of insertion/removing in the middle of array, this could be improved by using better data structure • Consider using Linked List • Date: 19th September 2023 • File path: src/utils/insert-foo.ts • File path: src/utils/remove-foo.ts

🔥 9.4 App is using a lot of memory

Description	Overusing immutable data structures can lead to a lot of junk memory being created.
Business consequences	The more memory and CPU applications are using there's a higher possibility for bad UX due to performance.
Suggestions	Stop using spread operator in reduce

Technical details

- You’re having a reduce that runs a lot of times, on each iteration it spreads the result creating a lot of junk memory
- Consider using non-immutable data structure
- Date: 19th September 2023
- File path: src/utils/bar.ts