BEJAMAS_

Technical Audit for Acme, Inc.

Last update November 27, 2023

We're taking a close look at a made-up company called Acme. Even though Acme isn't real, the problems we're digging into are based on real-world challenges.

React frameworks like Next.js and Gatsby have been around for 5 years. Companies using them from the start have run into issues with how their code is built, growing their platforms, and dealing with tech debt. We're here to spot those big problems and develop clear, doable solutions.

Authors

Dave Green

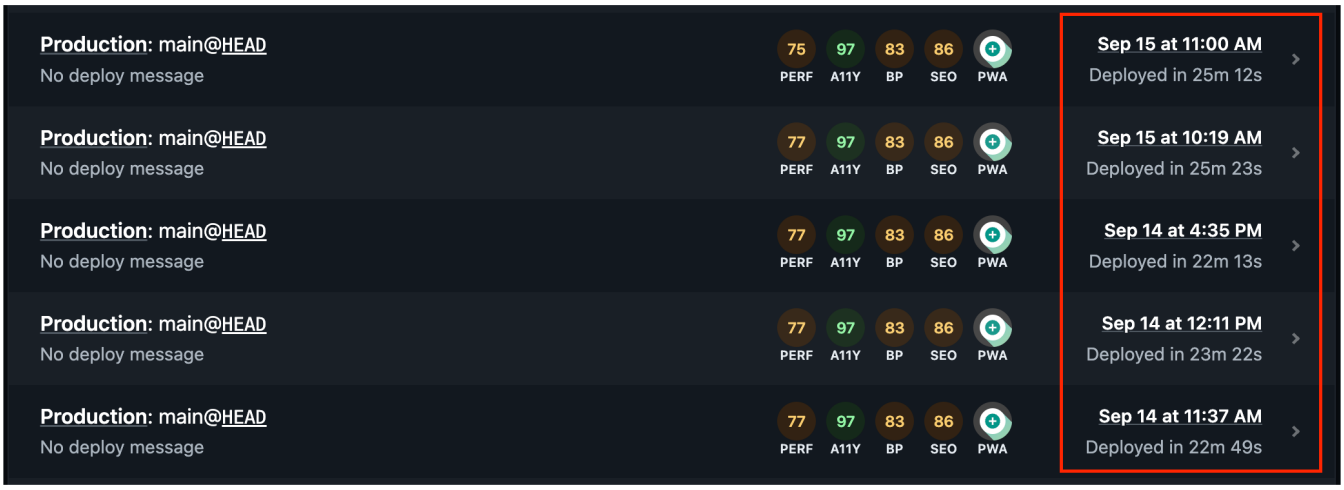
Software developer specializing in web development. Lover of JavaScript and modern web architecture. In a constant state of forward motion.

Aren Hovsepyan

I'm passionate about technology and data-intensive applications. I thrive at architecting and building scalable, consumer-focused products with throughput for hundreds of thousands users. I'm also keen on learning more about new, cutting edge technology solutions, such as crypto ecosystems and database storage systems for large scale applications.

Build Performance

Acme's current build times are concerning, often ranging between 20-25 minutes. For a modern web application, these durations are on the higher side, which can impact the speed of deployment and the ability to quickly roll out new features or critical fixes.



One of the primary reasons for the extended build time seems to be the data sourcing phase. Acme's application fetches data from multiple sources during the build process, and not all of these sources have been optimized for rapid data retrieval. This can lead to significant bottlenecks, especially when dealing with large datasets or when external data sources experience latencies.

Moreover, the application's codebase has grown over the years, and not all parts of it are efficiently structured. Some legacy code modules are processed during every build, even though they are no longer relevant or have been deprecated. This unnecessary processing adds to the build time.

Another contributing factor could be the lack of parallel processing during the build. Modern build tools allow for certain tasks to be run in parallel, which can significantly reduce overall build time. Acme's build process appears to be largely sequential, which means it doesn't fully utilize available resources.

To address these issues, Acme could consider:

Optimizing Data Sources: Reviewing and optimizing the data sources used during the build can lead to significant time savings. This could involve caching frequently accessed data, optimizing database queries, or even reconsidering the need for certain data sources.

Refactoring the Codebase: By cleaning up and refactoring the codebase, Acme can ensure that only relevant modules are processed during the build. This would involve identifying and removing deprecated code and ensuring that the remaining code is efficiently structured.

Parallelizing Build Tasks: Modern build tools like Webpack and Rollup allow for tasks to be run in parallel. By restructuring the build process to take advantage of this, Acme can ensure faster build times.

By addressing these areas, Acme could potentially reduce its build times by a significant margin, leading to a more streamlined and efficient deployment process.

Runtime Performance

The current runtime performance of Acme's application presents some challenges, particularly noticeable during key user interactions. While the application generally loads quickly on first visit, subsequent interactions such as navigating between pages, submitting forms, or loading dynamic content show noticeable lags.

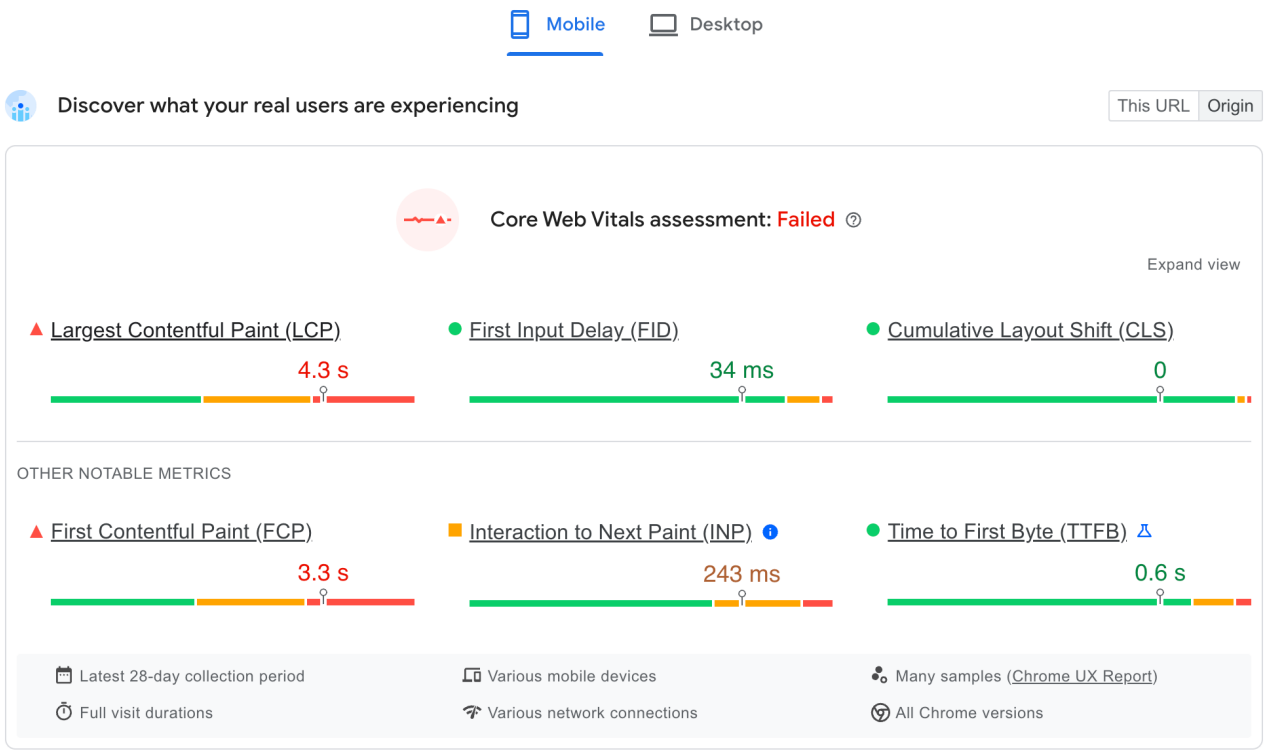
These performance hitches can be attributed to several factors. First, the application seems to be making redundant API calls. For instance, data that has already been fetched once is often re-fetched during the same session, leading to unnecessary wait times for users. Efficient caching strategies can be employed to address this.

Secondly, the application's frontend does not seem to be optimized for efficient rendering. Some parts of the UI experience jank, especially during animations or transitions. This can be due to inefficient JavaScript execution or non-optimal use of CSS. Leveraging browser developer tools can help pinpoint specific bottlenecks.

Lastly, the use of older or non-optimized third-party libraries could be adding to the runtime overhead. Periodically reviewing and updating these libraries can lead to substantial performance gains.

Web Vital Metrics

Web Vitals have become a cornerstone of evaluating web performance, especially from a user's perspective.



For Acme, some of these metrics show room for improvement:

- **First Contentful Paint (FCP):** This metric, which indicates the time it takes for the first piece of content to render on the screen, needs improvement. Slow FCP can be a result of large stylesheets, non-optimized images, or server latencies.
- **Time to Interactive (TTI):** Currently satisfactory, this metric indicates the time it takes for the page to become fully interactive. While Acme's TTI is decent, there's always room to enhance by deferring non-critical JavaScript or optimizing main thread work.
- **Speed Index:** This is below average for Acme. Speed Index measures how quickly the content of a page is visibly populated. Techniques such as lazy loading offscreen images or prioritizing visible content can be beneficial here.
- **Total Blocking Time (TBT) & Largest Contentful Paint (LCP):** Both these metrics can be improved by optimizing the critical rendering path, reducing server response times, and serving compressed assets.
- **Cumulative Layout Shift (CLS):** Minimal shifts are observed, which is good. However, ensuring that images have set dimensions and ads or embeds are dynamically loaded can further stabilize this metric.

Deep-dive into code

Project structure

The folder and file structure is not organised in a consistent way making it challenging to understand the architecture:

- Assets (images, pdfs, ...) are not stored the same way across the application:
- Global assets are stored in the static folder.
- When assets are only used for a single page, they are stored within a folder labelled with the route.
- When used across multiple sub-pages within the same category route, they are stored directly at the root category folder. This leads to inefficient development as there are many files visible when navigating the structure.
- Homepage specific images are located directly in the src folder.
- Some folders in src only contain assets like podcast, legal and get-started. These match up with HTML pages in src, but follow the same architecture pattern as routes. This will become more and more confusing as the application scales.
- Mixing HTML, styles and JS (apart from Nunjucks template syntax) together has led to large files that are difficult to work with. For example: src/about/index.html. As a result the codebase becomes incredibly difficult to maintain.
- Content is mostly created with HTML files while at other times, Markdown files are used.
- There is a lot of custom CSS especially for handling layouts. It's good that you have created CSS variables, but they do not cover nearly enough.

Images

Below are some examples of common issues we found:

- Non svg images are mostly using the webp format which is good, but there are further optimisations needed. For instance, adjusting image dimensions and

quality to reduce file size makes a significant difference to performance and passing core web vitals.

• In some cases, static styles are inlined and specific to the current images. The layout styles are overly complex and the combination would not scale well when trying to change or add more logos.

• We spotted that on the careers page the hero image is using an unoptimised format and the size of the image is massive. Other images on the page are also not sufficiently optimised.

• Additionally, hero images should be preloaded so the browser knows about them before parsing the HTML and can begin loading them as soon as possible.

• The blog is still loading images from WordPress, presumably the previous implementation. The images are not using optimised formats either:

Infrastructure Analysis

Acme's current infrastructure, while robust, exhibits complexities that can be streamlined. The multi-tiered approach, involving numerous third-party services, can sometimes lead to redundant processes and increased operational costs.

For instance, Acme's reliance on multiple cloud providers for different tasks (like storage on one and computing on another) might be leading to data transfer costs and latency issues. A consolidated approach, using a single cloud provider's suite of services, can simplify the architecture and reduce costs.

Additionally, the deployment process involves several manual interventions, making it prone to human errors. Implementing a comprehensive CI/CD (Continuous Integration/Continuous Deployment) pipeline can automate many of these steps, ensuring consistency and reducing the margin for error.

By revisiting the infrastructure design, Acme can not only achieve operational efficiencies but also realize significant cost savings in the long run.

SEO Issues

Search engine optimization is paramount in today's digital landscape, and while Acme's platform is reasonably optimized, there are evident gaps. For starters, there are missing meta tags across various pages. These tags play a crucial role in how search engines interpret and rank content. The absence of proper meta descriptions, titles, and alt tags for images can hamper visibility in search results.

Furthermore, Acme's sitemap appears outdated. An up-to-date sitemap helps search engines crawl and index content efficiently, ensuring all pages are considered. Regular audits and updates to the sitemap, especially after major content updates, are imperative.

Mobile responsiveness is another area of concern. With a significant portion of users accessing websites via mobile devices, ensuring an optimal mobile experience is no longer optional. Some pages on Acme's site exhibit layout issues on smaller screens, which can affect both user experience and SEO rankings.

Lastly, page load speeds, especially on mobile networks, can be optimized further. Compressing assets, employing a Content Delivery Network (CDN), and optimizing scripts can significantly enhance load times, benefiting both users and SEO.

Solutions

- Meta tags can be improved to include images and follow good SEO practices such as not using very long titles as is currently done for blogs.
- Structured data helps search engines understand what your site is about. This can lead to your site being included in more search results.
- As you can see in the image examples above, some are lacking alt attributes, or the quality of descriptions is poor, which hurt both SEO and accessibility.

• It looks like something went wrong when defining aria-labels for links that open in a new tab. The use of (opens in a new tab) is ok, but it should not be preceded by undefined. A better approach is actually to not use an aria-label in this case at all since screen readers should use the link text. We can then follow the text with the “opens in a new tab” instruction that is visually hidden.

• Blog posts are not using the accessibility technique above for links that open in a new tab.

Improvement Ideas

Acme's digital platform has evolved over the years, but with the rapid pace of technological advancements, it's essential to stay ahead. One immediate suggestion is to update to the latest versions of libraries and frameworks. This not only provides performance benefits but also ensures that the platform is secure against known vulnerabilities.

Continuous integration and deployment (CI/CD) is another area Acme can delve into. Automated testing and deployment pipelines can drastically reduce manual efforts, ensure code quality, and expedite feature rollouts.

Moreover, investing in training and upskilling the development team can pay dividends. Familiarity with the latest best practices, tools, and methodologies ensures the team can tackle challenges efficiently and innovatively.

Technology Evaluation

Acme's current technology stack serves its purpose, but as with all tech stacks, periodic evaluations are beneficial. For instance, while the current frontend framework provides a solid foundation, alternatives like Next.js offer enhanced developer experience, built-in features, and potentially better performance. Conducting a detailed pros and cons analysis for such options can guide informed decisions about potential migrations or integrations.

Migration suggestions

Given Acme's growth trajectory and evolving business needs, it might be prudent to consider infrastructure migrations. Solutions like AWS Amplify or Vercel provide scalable, performance-oriented platforms tailored for modern web applications. Such platforms offer built-in CI/CD, global CDNs, and serverless functions, potentially simplifying infrastructure management and scaling.

Proposed solutions

Better documentation

We should consider improving documentation as a necessary step rather than an optional one. Having better documentation for each part of the project is crucial for any new developer. This can help avoid the re-emergence of the issues noted and can assist developers in understanding the project's structure and architecture. This documentation can be included in the repo or presented in a small website for internal usage.

There are the beginnings of some documentation in src/meta, but there needs to be more comprehensive documentation covering each aspect of the project. We can create more structured docs for:

- Website architecture
- Design system (UI) - we can add Storybook
- Content structure
- General rules (branching strategy, commit strategy, code owners, etc.)

Images

11ty provides an image plugin we can use to ensure all images are properly optimised for both desktop and mobile.

Ideally, all images should be stored in the same location, either locally or using a digital asset management (DAM) platform like Cloudinary.

SEO and accessibility

We can improve on SEO by

- enhancing meta tags,

- adding structured data,
- and always using good descriptions for image alt tags which is important for accessibility as well.

Accessibility in the site is pretty good, but not consistent. The architecture suggestions above can help to avoid this issue.

Conclusion

In conclusion, the key issues with the current project revolve around

- a lack of organisation and clarity in code architecture;
- inconsistency with content creation methods, accessibility and styling;
- insufficient image optimisation;
- and a lack of documentation.

These issues have significant impacts on the efficiency of the development process, the reusability of components, general quality and performance of the site, and the onboarding of new developers.

A key goal for us would be to significantly enhance the project's maintainability and scalability.

The proposed solutions align with best practices and will be able to handle current and future needs of the project, making the codebase more robust and flexible for future developments.