Enhancing SEO Compliance of JavaScript Web Frameworks through Automated Testing inside CI/CD Pipelines*

Ján Ágh[1,*,†], Irina Makarova[1,*,†] and Pavle Dakić[1,2,*,†]

[1]Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia

[2]Faculty of Informatics and Computing, Singidunum University, Danijelova 32, Belgrade, Serbia

Abstract

This article aims to incorporate search engine optimization validation into the development process of JavaScript-based web applications, thus connecting two sides of optimizations, namely the selection of appropriate practices and their correct implementation. We summarize strategies for achieving search engine optimization (SEO) compliance in single-page applications (SPAs) and propose two distinct testing solutions aimed at addressing SEO concerns in modern web apps. Both approaches are incorporated into CI/CD pipelines, making it possible to enforce existing SEO strategies. The issue of visibility in search engines is especially characteristic for JavaScript-based frameworks. This is because of the way they load and render their content, are mostly unable to achieve satisfactory SEO scores and rankings. Due to the ever-increasing popularity of these frameworks, it is essential to identify and compare available optimization techniques to address this subject adequately. Here, we take a closer look at the SEO-friendly features of Nuxt.js and Next.js frameworks and evaluate their capabilities using two mock e-commerce websites as examples. To fulfill our second objective, an approach to continuous SEO testing is presented. We discuss the results of similar CI/CD pipeline setups using distinct technologies and tools, in particular Jenkins and GitHub actions. The results show that by integrating specific tests focused on SEO into CI/CD pipelines, developers can identify crucial shortcomings of their product and eliminate them before they cause harm in the production environment.

Keywords

Search engine optimization, Optimizing JavaScript for SEO, SEO compliance testing, Automated testing in CI/CD pipelines, Deployment automation

1. Introduction

In today's world, the concept of search engine optimization (SEO) is vital for receiving desired amounts of Internet traffic. It may also give valuable insight into the usability of a website. Generally speaking, if the indexability of a website suffers, it probably fails to offer a great customer experience as well. By optimizing certain aspects like the selection of suitable descriptive keywords or enhancements regarding content structure and loading speed, a website's ranking inside a given search engine can be drastically improved. While it is not necessarily hard to achieve these results with traditional websites (MPA) utilizing server-side rendering (SSR), frameworks built on top of JavaScript with widespread usage of client-side rendering (CSR) complicate the process [1, 2].

According to the Stack Overflow developers survey of 2023, frameworks such as React, Vue.js and Angular are among the most popular web technologies used by professional web developers [3, 4]. Their popularity indicates abundant presence of single page applications (SPAs) created using these technologies. The most visited social media platforms utilize JavaScript frameworks due to the numerous benefits they provide, both to developers and end users [5, 6]. Even though Angular was developed

SQAMIA 2024: Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, September 9–11, 2024, Novi Sad, Serbia

*Corresponding author.

†These authors contributed equally.

xagh@stuba.sk (J. Ágh); xmakarovai@stuba.sk (I. Makarova); pavle.dakic@stuba.sk (P. Dakić)

0009-0009-6916-6547 (J. Ágh); 0009-0006-3645-7987 (I. Makarova); 0000-0003-3538-6284 (P. Dakić)

more than a decade ago, search engines' support for SPAs is still not extensive. At the moment, there is no guarantee that an SPA would be processed correctly and timely by the search engines [7]. To mitigate SEO issues in SPAs, a new generation of frameworks emerged with Next.js and Nuxt.js that belong to this category [5]. These frameworks offer a variety of rendering methods and SEO-friendly features that simplify optimization for search engines while preserving the benefits of the SPAs architecture [8]. By strategically combining different rendering methods, these frameworks ensure that the page loads quickly and does not appear empty to the crawlers [1]. Although modern JavaScript frameworks simplify achieving high SEO scores, prompt detection of SEO issues remains complicated, regardless of the type of web application and scaling options on Kubernetes [9, 10].

Achieving desired SEO without maintaining good coding practices is impossible; despite that, in practice, teams dealing with development often work separately from the SEO specialists. It is not unusual for businesses to conduct an SEO audit after they encounter problems with online visibility. The cost of late detection of issues with SEO may be high, depending on the severity of the problems and the promptness of their identification. An efficient approach would be to evaluate compliance with the key SEO practices continuously, during the development stage [11].

Due to the growing demand for fast and secure delivery of new versions of software, platforms for continuous integration and continuous deployment (CI/CD) are gaining attention. The abilities to speed up, facilitate and automate code integration and deployment to customers have made CI/CD a far better choice than previously used monolithic methodologies [12, 13, 14, 15]. In addition to the already mentioned qualities, these platforms also provide options for defining targeted testing stages aimed at domains such as structural code integrity or security vulnerabilities, to name a few [16, 17]. Since having a CI/CD pipeline in place is becoming increasingly attractive for teams of developers, integrating SEO tests into the pipeline provides a solution that would be in line with the current software development trends.

The paper is organized as follows: the introduction section covers general information about the research and the problem we are trying to solve, materials and methods section state the objectives, the literature review section provides a brief theoretical background regarding relevant domains, results section describe and summarize the obtained solutions with the developed solution, analysis and discussion section examine possible limitations of the utilized approach and outline the main findings, and conclusion section contain the main observation and information.

2. Materials and Methods

The research in this paper was carried out using research questions, which are further detailed and defined in the section research Questions and objectives. Materials were obtained by searching published literature inside scientific databases as well as analyzing reputable online sources. The selection was made based on their relevance to the research topic. In the article, a small number of older sources were used, which have direct relevance to our research. The investigation utilized research questions to better understand the issues related to search engine optimization.

2.1. Motivation

Since modern JavaScript frameworks are becoming increasingly popular among web developers, the challenges brought by the need to be SEO compliant have to be addressed. Integrating SEO compliance testing into CI/CD test pipelines presents the first step how to discover possible problems and would prevent these problems from entering the production environment. Additionally, the connection between SEO compliance testing and CI/CD pipelines has not been comprehensively addressed in the literature, and therefore, more research has to be done to bridge this knowledge gap.

2.2. Research Questions and Objectives

In this article we examined in detail the relevant literature to provide answers to the research questions listed below:

1. What qualities do websites have to have to be SEO compliant?
2. Do all SEO factors affect the visibility of a website equally?
3. What are the ways to make websites using JavaScript frameworks more SEO compliant?
4. What are the existing approaches to testing SEO?

The answers are incorporated into the content of the Literature Review section. Based on the identified research gaps, two objectives of this paper were formulated: to evaluate SEO features of Next.js and Nuxt.js frameworks, and to create a continuous SEO testing solution.

The details of the developed testing solutions and SPAs are described in the section Architecture of the Solution.

3. Literature Review

This section briefly summarizes current knowledge in domains related to the issues addressed in the research topics. When discussing SEO, the emphasis is on commonly acknowledged principles, however the obstacles encountered in SPAs are also discussed. The literature review is divided into the following sections: single page applications, Next.js, Nuxt.js, search engines and optimization practices, and testing SEO compliance for Web applications.

3.1. Single Page Applications

Since the introduction of the XHR API, which allowed data to be sent and received asynchronously, early SPAs began to emerge and evolve gradually [2]. In one of the earliest studies regarding their architecture, an SPA web interface is defined as "composed of individual components which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action" [18]. From this definition, it can be inferred that the core difference between MPAs and SPAs relies in the way the communication between the client and the server takes place. When the most prominent JavaScript frameworks were introduced, CSR continued to be the primary rendering mechanism utilized in SPAs. Despite CSR being the main reason for the enhanced UX of such applications, this rendering approach caused problems with SEO, damaging the visibility of such websites [2]. These issues lead to development of a new generation of frameworks with extensive SEO considerations in mind. Due to poor search engine support of CSR, one significant advancement is support of novel rendering methods and their combination within the same page [19].

3.1.1. Next.js

The framework offers four rendering methods and allows to apply them on a per-component level as well as for individual pages. Table 1 compares available rendering modes based on a few selected properties that influence SEO, user experience and the development process itself, while highlighting similarities and differences between them.

Another benefit of Next.js is its suite of custom components with enhanced SEO features. One notable example is the `<Image/>` component substituting <img> element, which automatically resizes assets depending on viewport dimensions, prevents layout shifts and reduces page load time by means of lazy-loading and conversion of images to modern formats like WebP [22]. Another beneficial component is <Link/> which is an integral part of the Next.js framework. This component is used to enhance application performance for the client, while preserving the default behavior for the crawlers.

Apart from providing components, Next.js offers an intuitive way to define metadata for individual pages and add relevant keywords in the <title>, <meta name='description'> and other tags by retrieving

Table 1
Comparison of rendering techniques available in Next.js framework. Abbreviations: client-side rendering (CSR), server-side rendering (SSR), static site generation (SSR), incremental static regeneration (ISR). Source: information summarized from [19, 1, 20, 21].

Properties of rendering methods:	**CSR**	**SSR**	**SSG**	**ISR**
page generation	on request	on request	pre-rendering	pre-rendering
best suited type of content	dynamic	dynamic	static	dynamic
content updates	instantly	instantly	requires re-build	small latency
page load time on first request	slow	fast	fast	fast
server response time	fast	slow	fast	fast
load on the server	low	high	low	high

them from the page's URL and external data as well. Ways for the dynamic generation of the sitemap.xml and robots.txt files are also available in Next.js.

3.1.2. Nuxt.js

Nuxt.js is an open-source web development framework suitable for both front-end and back-end operations. It acts as an extension over Vue.js in that, besides the traditional CSR capabilities, it also provides other specialized rendering modes, amongst them SSR, SSG and hybrid rendering, and a variety of innovative features that streamline the development process [23]. In essence, Nuxt.js brings similar functionalities for Vue.js as Next.js for React.

An extensive advantage of this framework lies in its wide range of modules for the simplification of various tasks, like the automatic generation of one or more sitemap.xml files containing every detected route inside an application, the on-demand generation of the robots.txt file and various useful plugins for testing frameworks [24].

As its the case for Next.js, Nuxt.js also comes with built-in components for optimizing images, hyperlinks and elements crucial for SEO. Besides these, other innovative features include its file-based routing system and the ability to automatically import the contents of certain directories into any location inside an application [25].

3.2. Search Engines and Optimization Practices

Search engines are the most important intermediary actor between the users and the content they are looking for; by filtering and ordering pages based on their relevance to the user's query, they act as a two-way gateway that connects content providers with end users. With crawler-based search engines, and Google in particular, having the dominant share on the search engine market, it is essential to ensure the websites are optimized for the bots and other components of search engines. Generally, three major subsystems assist in distinct aspects of a web search: crawler, indexer, and ranking algorithms [26].

Optimizing for crawling stage, at which pages are being discovered and re-explored, is in the case of MPAs straightforward, while SPAs may cause issues due to empty DOM structure and slower initial rendering [2]. Even though Google supports SPAs by employing a rendering queue, processing of client-side rendered applications may be delayed in comparison to MPAs [7]. Thus, usage of isomorphic JavaScript and combination of several rendering strategies are recommended as effective measures to optimize SPAs for crawlers [1].

Indexing includes the processing and analyzing of a page's content and structure, which in a positive scenario leads to storing the URL, along with its metadata, in a search engine's index database. Only after the page is stored in the index, it can appear in search engine results page (SERP) [27]. Optimization of MPAs and SPAs for this stage is similar and includes both technical considerations and content research [28, 1].

The final, ranking step is, contrary to the previously discussed stages, directly dependent on the entered query. According to Google, relevance of the page to the searched query is the key position determining factor. However, query-independent ranking factors do exist, and might play a significant role in rivalry situations [29]. Also, the ranking factors' weights are subject to updates.

SEO practices aimed at optimizing websites for one or more search stages are frequently categorized into three groups based on their target [28, 30]:

- **Off-page SEO**, concentrating on actions taken outside the website's structure
- **On-page SEO**, targeting various elements regarding a website's structure
- **Technical SEO**, being a subset of on-page SEO dealing with everything besides the content itself

While not all authors put technical SEO into a distinguished category, it is relevant to separate these practices in the context of our objectives in this article. Figure 1 summarizes some of the most prominent on-page SEO practices, which include, but are not limited to semantic HTML elements, internal linking structure, assets optimization, textual content with keywords, URL structure and proper metadata [30].

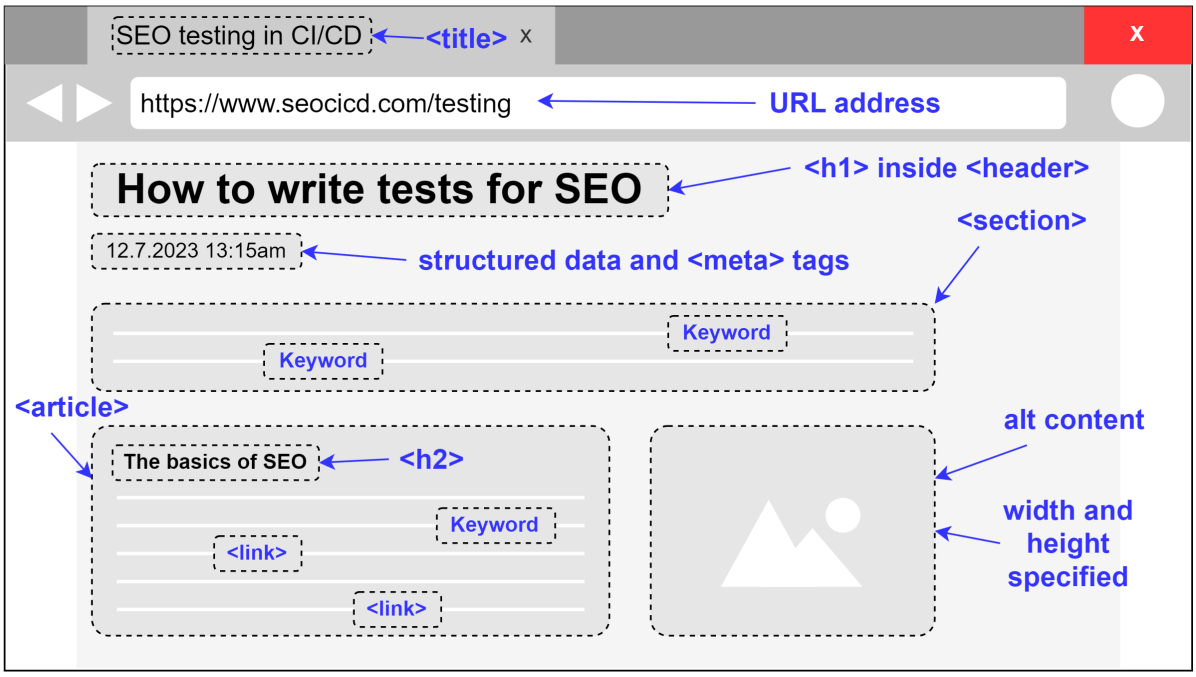

Figure 1: Structural information evaluated as part of on-page SEO; special attention is devoted to semantic elements, as indexers leverage them for guidance. Source: authors' contribution.

In the narrow sense, on-page SEO is concerned exclusively with the content, while technical SEO encompasses the rest of the above mentioned practices together with website's performance and security [31]. Thus, it can be concluded that technical SEO is the one most closely related to the web development process.

3.3. Testing SEO Compliance of Web Applications

DevOps has now become an industry standard, since it promotes automation and, at the same time, decreases the error rate in multiple areas of software development life cycle. Continuous integration and continuous deployment represent key DevOps practices that are widely used by developers' teams [13, 32, 33, 34]. Since DevOps encourages extensive testing inside CI/CD pipelines [35], incorporation of SEO tests, especially for the technical aspects of pages, presents a suitable and practicable approach to both evaluation of the website and prevention of SEO flaws in the deployed version.

To explore the state of the art, literature research was conducted in the domain of SEO. Four articles providing a comprehensive overview of SEO from the period 2019-2024 were analyzed [30, 26, 28, 31].

In regards to SEO testing Lighthouse was a frequently mentioned auditing tool [36]. Some authors presented their own SEO testing solutions [29]; and while the importance of continuous efforts in achieving SEO compliance was often emphasized, none of the analyzed studies reported SEO as a part of custom web application testing solution integrated into CI/CD pipelines.

There is, however, evidence that in industrial settings, this approach is utilized. The existence of platforms such as RankSense and Lumar indicates a demand for automation of SEO issues detection. These tools employ AI for testing and implementation of SEO practices to avoid problems in production and reduce error-prone manual effort. Lighthouse also comes with a version, named Lighthouse CI, that can be incorporated into CI/CD pipelines for continuous page auditing. Thus, to reduce the identified gap in scientific literature, we developed automated SEO testing solutions.

4. Results

This section provides details of the implemented testing solution based on the information compiled from relevant papers and online sources. In accordance with the collected sources, in the sections that follow, an architecture is proposed that looks at the possibilities of two SPAs Nuxt.js and Next.js frameworks. Through a defined basic structure, we try to test SEO complexities and the necessary requirements of implementing a fully functional and optimized website. Specific tests were related to some of the following aspects: image optimizations, performance, crawlability and indexability, etc.

4.1. Architecture of the Solution

Two SPAs were created in the e-commerce domain, each with the following goals in mind:

- to determine the extent of SEO capabilities of Nuxt.js and Next.js frameworks;
- to create and tailor SEO tests to the specifics of the developed application.

Both solutions have a similar CI/CD pipeline architecture with feature, development and main branches. In both cases commits to the main branch triggered the execution of SEO tests and application deployment on successful tests completion. Both web applications implement on-page and technical SEO practices from these groups: crawlability and indexability, performance, image optimizations, responsive design and mobile-friendliness, metadata, semantic HTML, URL structure and link validity, information architecture, trustworthiness and security, and keywords.

The tests were designed to cover widely accepted practices in the listed on-page SEO categories. The following sections highlight key features of the developed testing solutions.

4.2. SEO Compliance of Next.js and its Testing

The tools and frameworks utilized in this solution include:

- Next.js version 14.2.3 and React version 18.3.1
- Cypress version 13.9.0, Jest version 29.7.0, and the React testing library version 15.0.6
- Git, GitHub, GitHub Actions, and Vercel for deployment

The first set of frameworks was used to build the website, the second to perform SEO tests, and the third to set up the CI/CD pipeline with integrated custom tests. The solution was created and tested on a PC with the Windows 10 operating system and an Intel Core i7 processor.

For this solution SEO testing was divided into two separate stages, where component testing using Jest and React testing library occurred at the integration to development branch, and more extensive testing with Cypress was conducted before deployment. This method was chosen to detect faults with components that may be modified more frequently, such as a product card, and to speed up pre-deployment testing.

In regard to the website, pages intended for indexing were implemented to allow browsing products and categories. In terms of rendering methods, SSR and SSG were predominantly utilized; CSR was used

scarcely. During development, SEO practices were being continuously implemented and tested at the level of components, individual pages and files. Component level is mainly linked with semantic HTML, accessibility and image optimization; page level is coupled with rendering methods, internal linking, metadata and heading hierarchy; at the file level are special files such as robots.txt and sitemaps.

Next.js built-in capabilities were extensively utilized at each level; moreover, dynamic generation was prioritized for the metadata, special files and other SEO considerations.

4.3. Testing SEO Capabilities of Nuxt.js

Our second solution was created with the following frameworks and systems:

- Nuxt.js version 3.10.3 and Vue.js version 3.4.19
- Playwright version 1.42.1, Vitest version 1.4.0, and the Vue and Nuxt test utils libraries with versions 2.4.5 and 3.12.0
- Git, GitHub, Jenkins, Docker Desktop and DockerHub for deployment

The emphasis was on the development of a platform-independent solution by utilizing Docker for both the Jenkins CI/CD pipeline as well as for the tested application; for these reasons, DockerHub was chosen as the deployment environment. The solution was developed and tested on a PC with Windows 10 operating system and CPU AMD Ryzen 5.

The pipeline workflow executes the predefined Vitest component tests before deploying the dockerized application to a local testing environment for SEO end-to-end testing with the Playwright framework. The same testing steps were performed for both the master and development branches to be able to notify developers about the encountered SEO issues anytime a new feature gets merged, e.g. not just during deployment.

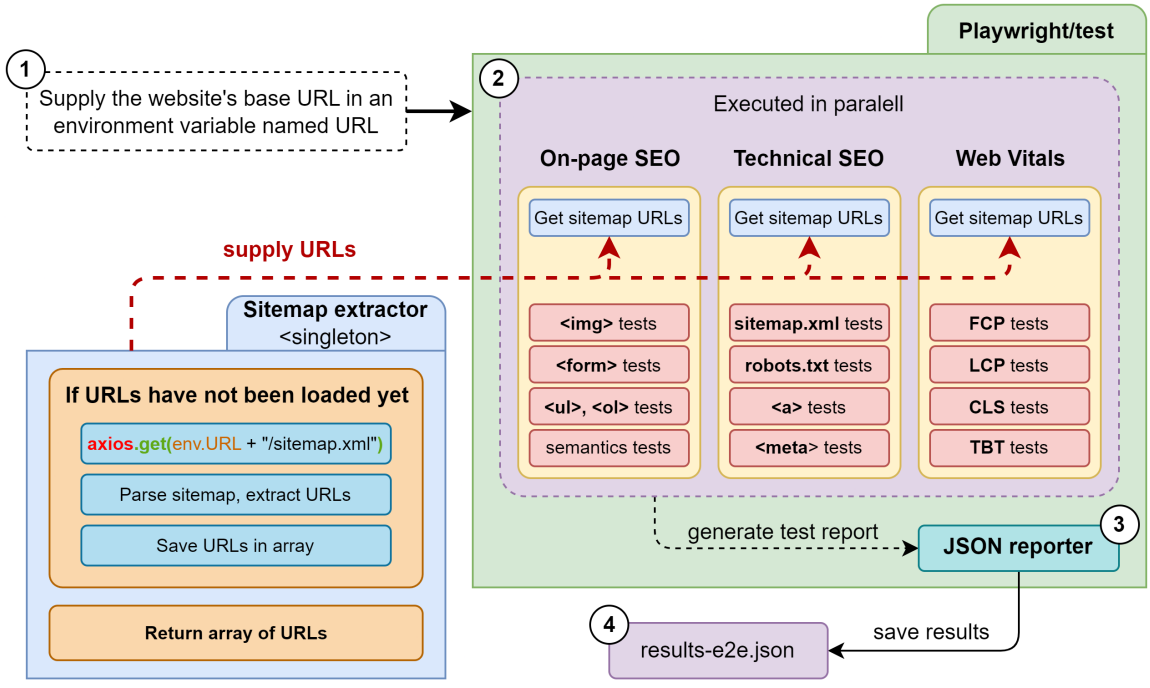

Figure 2: High-level visualization of our SEO testing module's structure. Source: authors' contribution.

An important contribution is our SEO testing module, visualized in figure 2, which is used inside the pipeline step responsible for the execution of SEO tests. This module is able to extract the tested application's pages from its sitemap.xml file and run three sets of SEO tests against them in parallel; the first set focuses on on-page SEO, the second one on technical SEO and the third one on the evaluation of relevant Web Vitals metrics. Deployment is allowed only if the tested application is able to pass every defined test.

To accurately test the capabilities of Nuxt.js, a combination of SSR and SSG were used for the application's pages together with built-in components for hyperlink definition and metadata generation. Extensive effort was also put into image optimizations and proper page structuring with semantic HTML elements.

4.4. Integration stage

The development of the web application was happening primarily on dev branch; specialized feature branches were utilized scarcely. Due to the fact that the project was not being developed by a team, the need for such branches was eliminated. However, the script for the pipeline was written in a way to support both branching configurations.

Tests for newly added features and pages were defined either in advance, or were committed at the same time to ensure every key component was tested before being merged with the functioning version of the application stored on the master branch. For the CI stage component tests written in Jest were run. Implementation of the complete CI workflow is shown in the Listing 1.

Listing 1: Defined workflow for continuous integration in Github Actions. Source: author's contribution.

```
1  name: CI to dev
2  on:
3  push:
4  branches:
5  - dev
6  jobs:
7  test-jest:
8  runs-on: ubuntu-22.04
9  steps:
10 - uses: actions/checkout@v2
11 - uses: actions/setup-node@v2
12 with:
13 node-version: "20"
14 - run: npm install
15 - run: npm test
```

The lines 2-5 specify events that will trigger execution of the pipeline. In this case, push event signifies that any commits to dev branch, including merging of feature branches, will trigger the provided workflow. Below the line 6 individual jobs can be defined by providing a name for the job, configuring environment and listing commands and actions to be executed sequentially; for our workflow one job named test-jest (line 7) was sufficient.

When the job is triggered, its steps will be executed on a virtual machine with Ubuntu Linix distribution version 22.04 (line 8). The job consists of two built-in actions that will checkout the latest version of the repository and download the requested Node.js version. Next, all the necessary project dependencies specified in the package.json file will be installed (line 14). Finally, all the Jest test suites will be run (line 15).

4.5. Evaluation

SEO compliance and performance of SPAs developed in Next.js and Nuxt.js was evaluated using Lighthouse tool version 11.6.0, which was run from DevTools. Both static and dynamic pages were audited; for multiple similar pages (e.g. product page), at most two representative pages were selected for the final evaluation. All the pages attained the highest score in the SEO, best practices and accessibility categories. In desktop mode, performance score always exceeded 95 out of 100 indicating great loading speed.

The developed continuous SEO testing solutions were compared with the Lighthouse tool. We identified several advantages of our custom solutions. Firstly, Lighthouse modifies test suite based on the content of the page, which may lead to missed SEO issues, while custom test suites always apply in their entirety. For instance, a website with no robots.txt can still get a perfect score in Lighthouse; despite that, it is advisable to provide this file in order to avoid wasting crawling budget.

Another advantage of custom SEO tests is their flexibility and extensibility. In our experience, Lighthouse is helpful for detecting basic SEO issues, but a perfect score is not indicative of high SEO compliance level. Most pages attained a perfect SEO early during development, before the majority of SEO practices were implemented. Custom SEO testing solutions allow to enforce compliance with the defined SEO strategy and highly specific requirements.

5. Analysis and Discussion

Other previous research has demonstrated the SEO compatibility of modern JavaScript frameworks. Here, we showed extensive SEO features of Nuxt.js and Next.js. Perfect Lighthouse SEO and performance scores of the created applications indicate that these frameworks indeed allow us to create SPAs compliant with all the major SEO requirements. Apart from that, many optimizations provided by the frameworks are built-in and require minimal effort from developers. In regard to performance, JavaScript bundling, code splitting and minification, along with numerous image optimizations are executed automatically by the frameworks. While Nuxt.js allows to select a rendering method for individual URLs, Next.js decides on the best rendering strategy based on the page's functionalities.

Due to the identified limitations of the Lighthouse tool it would be necessary to audit the applications by means of more advanced testing platforms to determine the true extent of the frameworks' capabilities. It would also be revealing to observe performance of the developed applications in the field and analyze the reports from Google Search Console.

Based on previously defined research questions we were able to come up with the following answers:

- RQ1: to get the full SEO compliance mark and 100% level, websites must focus on a variety of aspects. Starting with optimizing a page(s), it is critical to add proper metadata, such as a descriptive title and alternative tags, and to include semantic components when necessary. Aside from that, off-page optimization should not be overlooked, as the best results are obtained by integrating both SEO approaches.
- RQ2: developers can take a variety of ways to achieving proper SEO compliance using JavaScript-based frameworks, each with pros and cons. Dynamic Rendering was once a top solution for this issue, but many providers stopped supporting it after the introduction of hydration techniques. Today, developers are mostly urged to use frameworks that provide some means of employing hydration in conjunction with SSR or SSG, but work is additionally being conducted to provide even better alternatives.
- RQ3: Automation frameworks are the preferred option for SEO testing due to their ease of use and greater performance over DIY alternatives. Predefined tests can be included in any section of the pipeline. However, it is recommended to evaluate them prior to deployment, after other tests have been completed in both development and production environments.

The developed custom testing solutions proved to be as efficient as Lighthouse, and in some respects even more flexible and accurate. Available testing platforms, such as Jest, Cypress and Playwright, provide ways to implement SEO tests for all the key technical practices. However, to fully validate the developed testing solutions, it would be necessary to analyze how they perform in production.

The main limit might be illustrated by the leading search engines' continual adjustments to their needs and indexing algorithms. According to our findings, the bulk of visitors' early interactions with a website will have the most influence on their overall view of it, assuming SEO optimization is not done. That is why developers should try to reduce the impact of resource loading delays so that they do not severely degrade user experience. First Input Delay can provide important information for achieving this goal.

6. Conclusion

In this article we focused on the challenges that search engine optimization compliance poses to JavaScript frameworks, namely Vue.js and React, and ways how to overcome these problems by utilizing modern frameworks build on top of them; Nuxt.js and Next.js, respectively. The required qualities of SEO optimized websites were analyzed; a brief summary of the existing strategies of how to effectively implement SEO practices in SPAs during the development process was also provided.

The research of relevant literature was undertaken to identify gaps in the field of SEO testing; we observed a lack of scientific studies regarding continuous testing of SEO at the development stage, even though multiple authors and studies point to the importance of early detection of SEO issues. To positively contribute to this research domain, we implemented a custom solution that involved the creation of two e-commerce SPAs, one using Nuxt.js and the other one Next.js, each utilizing a distinct CI/CD pipeline architecture with integrated SEO testing capabilities.

Reinforced by the success of our practical implementation, we came to the conclusion that, if suitable combinations of optimization methods are implemented, SEO no longer puts JavaScript-based websites at a disadvantage. Moreover, we observed that integrating SEO tests into CI/CD pipelines provides an effective error-filtering layer able to catch otherwise hardly noticeable problems, of which SEO is undoubtedly one. The obtained results provide a good starting point for developing more robust continuous SEO testing solutions and further investigating the issue of avoiding problems with SEO in production.

Future research should focus on covering the possibility of creating binary libraries and a quick way of learning the content itself while maintaining the achieved SEO results. We feel that one method to achieve this would be to consider the capabilities of WebAssembly (Wasm), which provides a portable binary-code format, and the Rust programming language. With this, it is feasible to obtain secure code with efficient memory management on the server side, allowing for improved availability and servicing a larger number of users.

Acknowledgments

The work reported here was supported by Erasmus+ ICM 2023 No. 2023-1-SK01-KA171-HED-000148295, the Cultural and Educational Grant Agency of Slovak Republic (KEGA) Model-based explication support for personalized education (Podpora personalizovaného vzdelávania explikovaná modelom) - KEGA (014STU-4/2024), Scientific Grant Agency of Slovak Republic (VEGA) under grant No. VG 1/0675/24, National project "Increasing Slovakia's resilience to hybrid threats by strengthening public administration capacities", project code ITMS2014+:314011CDW7 and supported by the European Social Fund, finally co-funded by the European Regional Development Fund (ERDF) and the Operational Program Integrated Infrastructure for the project: National infrastructure for supporting technology transfer in Slovakia II – NITT SK II, co-financed by the European Regional Development Fund.

References

[1] K. Kowalczyk, T. Szandala, Enhancing seo in single-page web applications in contrast with multi-page applications, IEEE Access 12 (2024) 11597–11614. doi:10.1109/access.2024.3355740.

[2] A. Huotala, M. Luukkainen, T. Mikkonen, Benefits and challenges of isomorphism in single-page applications: Case study and review of gray literature, Journal of Web Engineering (2023). doi:10.13052/jwe1540-9589.2186.

[3] F. Cák, P. Dakić, Creating Feature Model for YAML Generator in CI/CD Pipelines with React Web Application, Springer Nature Singapore, 2024, pp. 529–539. doi:10.1007/978-981-97-3305-7_42.

[4] Stack overflow annual developer survey, 2023. URL: https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe.

[5] J. Vepsäläinen, A. Hellas, P. Vuorimaa, The rise of disappearing frameworks in web development, in: Lecture Notes in Computer Science, Springer Nature Switzerland, 2023, pp. 319–326. doi:10.1007/978-3-031-34444-2_23.

[6] M. Varela, L. Skorin-Kapov, T. Maki, T. Hosfeld, QoE in the web: A dance of design and performance, in: 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX), IEEE, 2015. doi:10.1109/qomex.2015.7148084.

[7] A. Pomaro, Seo for javascript: An experiment to test search engines, 2022. URL: https://www.oncrawl.com/technical-seo/seo-javascript-experiment-test-search-engines/.

[8] H. A Jartarghar, G. Rao Salanke, A. K. A.R, S. G.S, S. Dalali, React apps with server-side rendering: Next.js, Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 14 (2022) 25–29.

[9] F. Eyvazov, T. E. Ali, F. I. Ali, A. D. Zoltan, Beyond containers: Orchestrating microservices with minikube, kubernetes, docker, and compose for seamless deployment and scalability, in: 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), IEEE, 2024. doi:10.1109/icrito61523.2024.10522382.

[10] T. Golis, P. Dakić, Creating a Self-Service DevOps Platform for Black-Box Testing on Kubernetes, Springer Nature Singapore, 2024, pp. 345–355. doi:10.1007/978-981-97-3305-7_28.

[11] H. Batista, Catching seo errors during development using automated tests, 2019. URL: https://searchengineland.com/catching-seo-errors-during-development-using-automated-tests-322365.

[12] P. Dakić, V. Todorović, V. Vranić, Financial justification for using ci/cd and code analysis for software quality improvement in the automotive industry, in: 2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC), 2022, pp. 149–154. doi:10.1109/ZINC55034.2022.9840702.

[13] N. Govil, M. Saurakhia, P. Agnihotri, S. Shukla, S. Agarwal, Analyzing the behaviour of applying agile methodologies & devops culture in e-commerce web application, in: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), IEEE, 2020. doi:10.1109/icoei48184.2020.9142895.

[14] N. Hroncova, P. Dakic, Research study on the use of ci/cd among slovak students, in: 2022 12th International Conference on Advanced Computer Information Technologies (ACIT), IEEE, 2022. doi:10.1109/acit54803.2022.9913113.

[15] P. Dakić, I. Stupavský, V. Todorović, The effects of global market changes on automotive manufacturing and embedded software, Sustainability 16 (2024) 4926. doi:10.3390/su16124926.

[16] P. Dakić, M. Živković, An overview of the challenges for developing software within the field of autonomous vehicles, in: 7th Conference on the Engineering of Computer Based Systems, ECBS 2021, ACM, 2021. doi:10.1145/3459960.3459972.

[17] M. Marandi, A. Bertia, S. Silas, Implementing and automating security scanning to a devsecops ci/cd pipeline, in: 2023 World Conference on Communication & Computing (WCONF), 2023, pp. 1–6. doi:10.1109/WCONF58270.2023.10235015.

[18] A. Mesbah, A. van Deursen, Migrating multi-page web applications to single-page AJAX interfaces, in: 11th European Conference on Software Maintenance and Reengineering (CSMR'07), IEEE, 2007. doi:10.1109/csmr.2007.33.

[19] R. Hanafi, A. Haq, N. Agustin, Comparison of web page rendering methods based on next. js framework using page loading time test, Teknika 13 (2024) 102–108. doi:10.34148/teknika.v13i1.769.

[20] R. Gallego, Next.js rendering strategies: Ssr, ssg, and isr compared, 2023. URL: https://hybridheroes.de/blog/2023-05-31-next-js-rendering-strategies/.

[21] L. Robinson, A complete guide to incremental static regeneration (isr) with next.js, 2021. URL: https://www.smashingmagazine.com/2021/04/incremental-static-regeneration-nextjs/.

[22] S. Sasikumar, S. Prabha, C. Mohan, Improving performance of next.js app and testing it while building a badminton based web app, SSRN Electronic Journal (2022). doi:10.2139/ssrn.4121058.

[23] L. T. Kok, Hands-on Nuxt.js Web Development: Build universal and static-generated Vue.js applications using Nuxt.js, Packt Publishing, Limited, 2020, pp. 8–82.

[24] Nuxt.js authors, Nuxt modules, 2024. URL: https://nuxt.com/modules?category=SEO.
[25] L. T. Kok, Hands-on Nuxt.js Web Development: Build universal and static-generated Vue.js applications using Nuxt.js, Packt Publishing, Limited, 2020, pp. 83–249.
[26] A. Shahzad, D. W. Jacob, N. M. Nawi, H. Mahdin, M. E. Saputri, The new trend for search engine optimization, tools and techniques, Indonesian Journal of Electrical Engineering and Computer Science 18 (2020) 1568. doi:10.11591/ijeecs.v18.i3.pp1568-1583.
[27] T. Yang, A. Gerasoulis, Web search engines: Practice and experience, CRC Press, 2014, pp. 1–24. doi:10.1201/b16812-57.
[28] D. Sharma, R. Shukla, A. K. Giri, S. Kumar, A brief review on search engine optimization, in: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), IEEE, 2019. doi:10.1109/confluence.2019.8776976.
[29] D. Lewandowski, Web searching, search engines and information retrieval, Information Services & Use 25 (2006) 137–147. doi:10.3233/isu-2005-253-402.
[30] N. Makrydakis, Seo mix 6 o's model and categorization of search engine marketing factors for websites ranking on search engine result pages, International Journal of Research in Marketing Management and Sales 6 (2024) 18–32. doi:10.33545/26633329.2024.v6.i1a.146.
[31] W. Wahba, T. Barhoom, Seo: Improve website ranking based on competitors analysis, International Research Journal of Engineering and Technology (IRJET) 06 (2019) 782–786.
[32] P. Dakić, Importance of knowledge management for CI/CD and Security in Autonomous Vehicles Systems, 2024. URL: https://jita-au.com/index.php/2024/06/13/importance-of-knowledge-management-for-ci-cd-and-security-in-autonomous-vehicles-systems/. doi:https://doi.org/10.7251/JIT2401007D.
[33] A. Dhulfiqar, M. A. Abdala, N. Pataki, M. Tejfel, Deploying a web service application on the edgex open edge server: An evaluation of its viability for iot services, Procedia Computer Science 235 (2024) 852–862. doi:10.1016/j.procs.2024.04.081.
[34] P. Dakić, Software compliance in various industries using ci/cd, dynamic microservices, and containers, Open Computer Science 14 (2024). doi:10.1515/comp-2024-0013.
[35] T. Golis, P. Dakić, V. Vranić, Creating microservices and using infrastructure as code within the ci/cd for dynamic container creation, in: 2022 IEEE 16th International Scientific Conference on Informatics (Informatics), IEEE, 2022. doi:10.1109/informatics57926.2022.10083442.
[36] T. Heričko, B. Šumak, S. Brdnik, Towards representative web performance measurements with google lighthouse, in: Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC), StuCoSReC, University of Maribor Press, 2021. doi:10.18690/978-961-286-516-0.9.