Tuhina Das

ISM II

Research Assessment #6

Date: September 29, 2024

Subject: Full-Stack Software Development

Assessment: Research Assessment #6 - Comparing Webpage Rendering Methods

As I begin planning ideas for my original work, I have decided to turn the focus of my research to familiarizing myself with some of the more technical aspects of software development. I have noticed, in the months following the conclusion of my previous year's project, that some of the tools I used are now considered deprecated – one of these tools being Node Package Manager's (npm) create-react-app build environment, which has just recently been replaced in popularity by Next.js, a full-stack web development framework. Every developer tool has edge cases that define the limits of that tool and its configurations – in this case, Next.js has various rendering methods, each with strengths better tailored for different types of web applications.

There are three rendering methods defined by Hanafi et al. in their publication: server-side rendering, client-side rendering, and static site generation. From my understanding, there are two main types of websites: static websites, and dynamic websites. Static websites do not handle user data (such as logins), whereas dynamic websites do – it is the latter which I have centered my work around as an aspiring software developer. Static site generation (SSG) deals with the rendering of static web pages: the server hosting the website displays page data immediately. Client-side rendering (CSR) and server-side rendering (SSR), on the other hand, deal with the rendering of dynamic web pages involving user data; in other words, data must be

retrieved from a database in the application's back end and then rendered alongside the rest of the page. I was unfamiliar with the concepts of CSR and SSR before reading Hanafi et al. 's publication, which defined the two for me; however, one limitation I wish the article better addressed is the type of web applications better suited in theory for each rendering method. I could infer, however, that SSR would be suboptimal for applications requiring constant re-rendering, such as live chat applications or streaming applications.

Hanafi et al. investigate the efficacy of these rendering methods by analyzing the speed at which various pages load, using a testing method called the Page Loading Time Test. The Page Loading Time Test involves the usage of the network features menu in the DevTools panel of the three different web browsers utilized for this study. While I am familiar with DevTools, this is the first time I have heard of using the aforementioned network features menu – as I hone my skills in enhancing user experience, this may be a tool worth further investigation for future use. I am also increasingly fascinated by how internetworking theory is so intricately woven into development; my knowledge in this field will help me build applications better tailored to a user's network.

The results of Hanafi et al.'s study surprised me significantly. Prior to reading this publication, I was under the impression that SSG would take the least amount of time in any case to render a web page, since there theoretically should be little back-end activity from my understanding. While this was the case with Profile and Home Page rendering tests conducted in the study, the Resource Comparison on Home Page Testing and Request Comparison on Profile Page Testing showed vastly different results, with SSG far exceeding CSR and SSR in time. This helps me understand why SSG is relatively ineffective when compared to CSR and SSR in handling dynamic content: it takes far too much time for such applications to retrieve back-end

data in the first place. Furthermore, SSR demonstrated taking considerably more time to load and finish generating page content consistently across all tests, relative to CSR. I can safely infer, from these results, that it is optimal for developers to use CSR consistently unless SSR is absolutely required – this information will help me better optimize the speed at which my future project runs.

APA citation(s):

Hanafi, R., Haq, A., & Agustin, N. (2024, 3 13). Comparison of Web Page Rendering Methods Based on Next.js Framework Using Page Loading Time Test. *TEKNIKA*, *13*(1), 102-108. https://doi.org/10.34148/teknika.v13i1.769