Name: Rojauhn Noble 100849533

For my static to-do list application, I went with Next.js instead of 11ty since it provides more current development standards and greater flexibility—even for static applications. Although 11ty works well with templating languages like Nunjucks to create static websites, its main purpose is for static projects with a lot of information, like portfolios or blogs. However, even for static apps, Next.js offers a more dynamic, component-driven architecture that improves flexibility and maintainability. Even though the app is static in nature, it becomes easier to manage and scale over time by utilizing React components to construct modular, reusable parts like task items and filters.

Even in a static context, Next.js excels in performance as well. The speedy loading of my to-do list app is made possible by features like automatic code-splitting and static generation (SSG), which provide a snappy, quick user experience. Although 11ty is built for static builds, Next.js gives greater versatility by enabling both static generation and server-side rendering, which could be handy if I ever need to shift the app from static to more dynamic interactions in the future.

Furthermore, Next.js makes handling CSS and static assets easier. My app allows me to scope styles directly to individual components, which guarantees clear, conflict-free styling that will be simpler to update as the app expands. This modular approach to CSS is more efficient than the old global stylesheets commonly used in static generators like 11ty. Additionally, Next.js's build process is reliable and works well with contemporary workflows, making deployment and scaling simple—even for static applications. Ultimately, Next.js provides a more flexible and future-proof solution for my static to-do list app than both frameworks, even though both can manage static sites effectively.