**INFT3102 - Web Frameworks – Assignment 1**

**Documentation**

Brody Dentinger

100 561 604

## **Overview**

I successfully implemented Next.js as a static site replacement for 11ty. I met my proposal requirements, and even managed to add a few new features as well. I leveraged the out-of-the-box capabilities Next offers to statically build most of the assets on my website. In the context of this assignment, this would make it very easy for a non-tech literate person to simply edit the corresponding JSON file for any content updates. Next’s toolbox was very simple and effective to use for these purposes.

The documentation Next.js provided was extremely helpful. I began my project with the starter template provided at <https://nextjs.org/learn-pages-router/basics/create-nextjs-app/setup>. This “Get Started” pathway also provided lessons on static assets, dynamic routing, page-based routing, and much more of Next’s capabilities and basics. The majority of the techniques I employed were through a combination of official documentation, and examples provided by ChatGPT to understand how pieces could fit together in some scenarios.

## **Theme**

The SSG project I built is a personal developer portfolio showcasing my own projects for potential employers. The portfolio data consists of a project title, a brief description for the portfolio cards, a detailed description for each individual project, an array of technologies used, and an array of images that are showcased in a carousel for each project. The pages will be created at build-time by utilizing several out-of-the-box functions, namely GetStaticPaths(), and GetStaticProps(). This will improve the performance of the website by creating the pages at build time instead of request time.

This build process is automated with the command: npm run dev. This will start the next.js development server on localhost:3000 by default. Npm install may be necessary to ensure all dependencies are installed on your machine as well.

A screenshot of a computer

Description automatically generated

## **Technologies**

As mentioned, I used next.js for this project. The advantage of using next.js is that many of the technologies that were recommended for this project, such as Nunjucks and 11ty, are featured out-of-the-box with next, which simplifies and streamlines the development process. I will go deeper into the details below.

#### Static Site Generation

I chose Next because it has SSG capabilities built in. Additionally, this framework has many other capabilities built in, allowing for a very flexible and scalable project moving forward. This is important to me because I need a portfolio website for future job applications, and utilizing a newer technology will allow for constant improvements over time, as well as provide evidence of knowledge of recent technology stacks.

A screen shot of a computer program

Description automatically generated

#### Data and Templating

Data for each page will be stored in a corresponding JSON file. For example, portfolio project data will be stored in a JSON file called projects.json.

I used another out-of-the-box feature called dynamic routing for the creation of individual project pages. These paths are still creating statically at build time. By creating a page with [] around the name, Next automatically knows that this is a dynamically routed page. It will invoke the GetStaticPaths() function to return a list of newly created routing paths.

For myself, I utilized the id property in my projects.json to create a page/route for each project in that JSON file. Next will then also automatically invoke GetStaticProps() on each of the newly created pages/routes, which returns the data from the JSON as props, which were passed to the html templating of each project page.

I used components and layouts within Next, which allowed me to reutilize pieces of the UI whenever necessary. In cases such as the project image carousel, I used GetStaticProps() to statically inject an array of images as props into each project pages carousel component. The flexibility to utilize these props in many ways made this intuitive and simple. In my lab 1 design, I was not sure if I would be able to display each project on a separate page easily. As mentioned above, with dynamic routing this was quite easy, and I did end up including paginated projects.

Inside of my react components, I used JSX instead of Markdown. Markdown is possible to use, but again, I think it makes more sense to leverage next to the fullest and utilize JSX to build the component templates. Adding additional technologies will complicate the project, and I don’t see a distinct advantage of using markdown in this scenario.

A screen shot of a computer program

Description automatically generated

On the main portfolio page, I displayed all the portfolio entry in as a templated card. This leveraged the reusability of react components, in that as more projects are added to the portfolio, they can simply be added to the JSON.

I also ended up including some filters based on the technology included in each project. An advantage of having the data already organized in a projects.json file, is that the foundation for utilizing that data to filter the projects was already in place. To do this, Next (and I believe React as well), offers a useState() function, which allows for state manipulation and tracking out-of-the-box. Another reason why Next is quite elegant and intuitive.

#### Style

For my styling, I ended up using Chakra UI. This is a component-based UI library that is popular within the field. I wanted to learn a UI library, because thus far in this program, we have only used bootstrap. Bootstrap is convenient but not very flexible, and I consider it a bit “amateur” at this point. I wanted to push myself to branch out in this department. I leveraged Chakra in many of my components and JSX templates to take advantage of their custom tags. Many of their tags offer out-of-the-box accessibility and responsiveness. This means my webpage is reactive to different window sizing. I utilized the Chakra documentation to implement some basic components and tags into my work. <https://v2.chakra-ui.com/getting-started>. The official documentation was clear and easy to use.

Currently, because Chakra and the way Next organizes CSS was a bit of a learning curve for me, I ended up using a combination of global styling, component styling, and in-line styling. This is not ideal as it is disorganized. I want to consolidate my styling and take advantage of further Chakra features in the future. (Discussed more in future features section)

## **Page flow**

A screenshot of a computer

Description automatically generated

The main page is a welcome page, with a brief introduction heading, a horizontal navbar, and a social link component. The nav bar includes the home page, about page, and a portfolio page.

The portfolio page displays all projects from projects.json initially, and can also be filtered with the technology tags at the top of the page. This allows users to seek out projects that include a specific technology or language.

The about me page has some detailed and specific sections that are drawn from about.json. There are some quick navigation links that will scroll down to the section a user is more interested in. These are drawn from the JSON file as well, so when edits are made to about.json, this will update automatically.

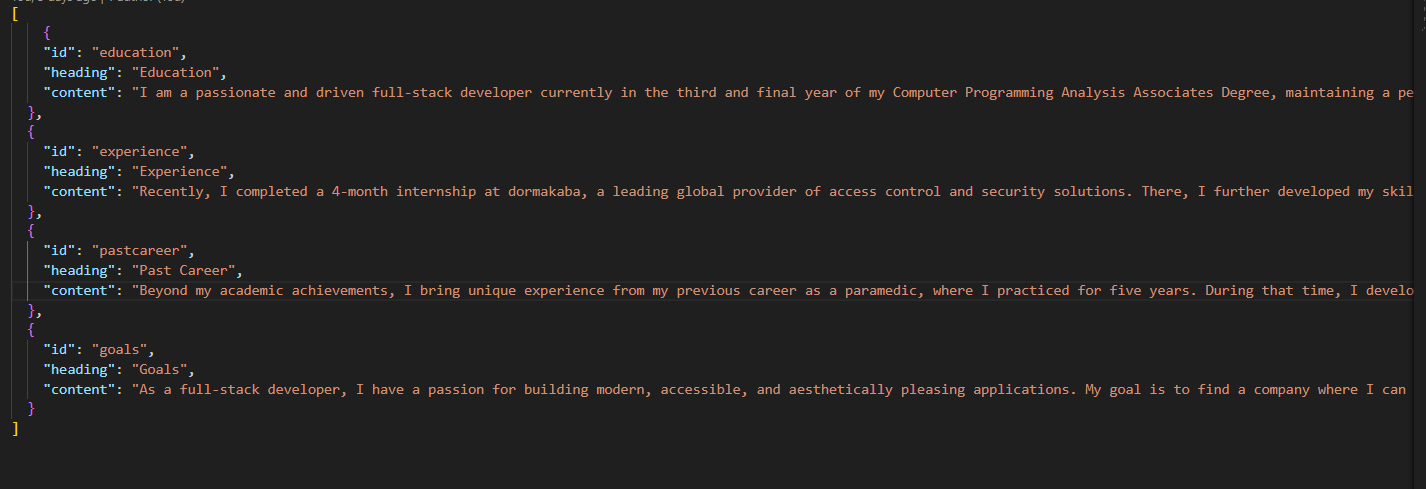
## **Maintenance**

As mentioned, future maintenance on this website could be done by either a developer or even a non-tech related person. This is one advantage to building a CMS-styled website.

All content for each page is tied to a related and aptly named JSON. For example:

* about page pulls data from about.json
* Home from homepage.json
* Navbar from navlinks.json
* And portfolio projects from projects.json

Any content that would need to altered, added, or deleted, can be accessed in these reader-friendly JSON files.



Changing the styling such as colours and font would require altering the CSS. Perhaps a future feature would be to implement theme, colour, and font pulls from a JSON as well. I imagine this is possible with Chakra themes. That being said, that also may be over-the-top considering this is not a full CMS service clone.

## **Future Features**

A contact page should be added at some point to display my contact information, and potentially include the capability to fill in text boxes and send the details to my email address.

I would still like to add Page navigation between projects. For example, having a “Next/Last Project” button to navigate to the next project, instead of having to return to the Portfolio page to select a different project.

I want to fully take advantage of React/Next Layouts in the future. I would like to implement shared elements across all pages, such as the Navbar and Socials.

As mentioned briefly above, I want to consolidate my styling and take advantage of further Chakra features in the future. I would like to make use of global “Themes” that Chakra offers to cut down on repeated styles. I would also like to have all other non-Chakra related CSS strictly in the styles files, and completely eliminate any page-specific CSS.

Additionally, I would like to implement an “infinite scroll” feature on my project page. I have heard that is a modern approach to pagination, and it would fit well with my theme. The filters being in place ensures users would still be able to narrow their search, even with many projects. I would also like to implement a search bar for specifically finding a project a user might have in mind. This could be done using the JSON data.