National Parks Showcase Documentation

Introduction:

This is a React-based project that's purpose is to showcase national parks dynamically. The idea is that this website would show a handful of national parks along with information about them. The project has responsive navigation bars, interactive features like carousals and accordions, and data that is contained in one array and distributed throughout the project. For the sake of making a fluid website. This project uses 3 different pages and has six different components. The home page is the only page with the components that carry data.

Page Routing:

The page routing is done through react-dom. Which defined roots using the “Routes” keyword that pages can be put on. The root “/” is pathed to the home page. This is also the default. With “/about” pathed to the about page and “/contact” pathed to the contact page. The navbar uses these roots to switch to the root specified using the Link keyword. The Outlet keyword is used to store information from other routes like page name.

About Page & Contact Page

These pages both have a static header, footer, and navbar. Both pages are heavily styled from tailwind.css templates. After adding multiple sections, changing font size, colors and removing wanted features they were then completed with dummy text from AI. The contact page will remove text after submitting but it won't go anywhere.

Data & React Prop:

The data for this project comes from parksData.js. It contains an array that is imported to the Home Page. That page then uses the data as a prop hook and all or parts of the array are sent into the Accordion, Carousal, Modal, and Tool tip Components. It’s then used to display that data in a variety of ways.

Home Page:

The home page is the main page of this project. It uses react props to distribute national data across every component that uses it. The home page first has a simple section that explains what a national park is. Then a section where the user clicks the Learn more buttons there is taken to a modal that links them to a website of a specific national park. Then if they scroll down, they find a connected carousel and accordion. The carousel scrolls through national parks and the accompanying accordion when click will display

information related to the park the carousel is on. The home page has a header and footer just like the other two pages.

Components

1. Carousel Component

The Carousel component is the main key feature of this project. This displays an image, two buttons, and the park name. When the Carousel is first loaded in it uses a react hook set to the first image to the first index and to close the accordion. Then when the arrows are clicked it will set the next image based on the if the back or forward button is clicked. If back the array goes to the previous id or last id and reverse for forward. This component does require the image id to be in the array and a react hook to remember the state of the carousel.

2. Accordion

Unlike the previous assignments, this project has a dynamic accordion. This accordion needs to keep track of data so instead of being defined in the home page like before. It is defined in the carousal component. The accordion has three pieces of information that it gets from parksData.js. The location of the national park, how it was established, and prominent features. This information is passed from the carousel, so it's updated when the carousel arrows are pressed. The accordion closes on any arrow click to keep the website performance up.

3. Modal Component

Each national park has a website in its array. The idea of the Learn More Modal is that one of these parks is selected and then its website and name are linked to it. In this one the chosen park in Yellowstone. Then that park's data is propped into a Modal and the Modal has the text "More about {park.name}" and that texts linked to that park's website. The modal opens or closes on button press and uses stopPropagation to prevent other interactive features from being used until the modal is closed.

4. Tooltip.

When the header above the carousal is hovered over it has a tooltip with instructions on how to use the carousal. The home page sends two things to the tooltip. One is the instructions the tooltip displays and the other is the header itself. As both as propped into

the Tooltip the header is called the child class. The tooltip itself makes the instructions pop up when it's hovered over.

5. Header & Navbar

The header contains a title that is just text styled in tailwind css and a responsive navigation bar. The navigation bar can access any page it’s linked to via react-router-dom. It also has a feature where it folds into a hamburger menu when the screen gets small enough. By using a media query after the screen reaches a certain size the nav bar disappears and hamburger menus appear. Then the navbar reappears when the hamburger menu is clicked but the navbar display is a column until the screen is increased again. This is done in CSS (not tailwind).

6.Footer

The footer simply displays tailwind CSS styles text about copyright.

Website Enchantments:

Responsiveness

It used tailwind's classes that change the size when breakpoints are reach like md (medium breakpoint) and sm (small breakpoint).

A hidden hamburger menu that appears only if the screen is small enough and displays the nav bar vertically for a better small screen experience.

Performance

This project uses stopPropagation and conditionals to make the website perform better. StopPropagation makes the other parts unable to work until the modal is close which helps performance and, the accordion close automatically when sliding through the carousal which did increase the speed.

The other aspect that helps performance was the react hook. Use state was used throughout the project which made it easier for my servers to remember if something is in use without needing to check it again.