LFG Take-Home Assignment

> Next.js Data Fetching with PokeAPI

In this assignment, you will create a Next.js project that demonstrates your ability to set up a new application, fetch data from an external API, and render that data on the client.

This assignment is designed to assess your skills in web development and your familiarity with Typescript and the latest Next.js features.

It's alright if you aren't already familiar with these technologies; use this opportunity to learn them!

Instructions

Setup

1. Create a GitHub repository for this assignment.
    - Include a `README.md` file with a brief description of this assigment.
    - You may also use this README to outline any design decisions or assumptions made in this assignment here.
    - Ensure that the repository is set to public.
2. Initialize a Next.js project using `create-next-app`.
    - You must use TypeScript and the `/app` directory.
    - You may use TailwindCSS for styling.

Tasks

We will be using PokéAPI for these tasks. No authentication is required for this API, and only the `GET` method type will be used with the `/api/v2` endpoints.

Data fetching

This task assesses your ability to use the `fetch` API and server-side rendering feature of Next.js.

1. On the root page of your app, using a server component, fetch and render the names of all 1292 Pokémon (at the time of writing).
2. Using dynamic routing, create a page for each Pokémon, and render its sprite.
3. Add a link from these dynamic pages back to the root page.
4. Back on the root page of your app, add a link for each rendered Pokémon name to its page.

Intermediate: Finding Pokémon

For this task, we will update the root page of the app. You may opt to implement this feature on the server (using query params) or on the client (using `useState`)—which you choose is up to you.

Using a client component, create a search bar which filters Pokémon on the root page by their name.

Advanced: Creating a Pokémon team

For this task, we want to allow the user to create a team of **up to 6 Pokémon**.

You are not required to required to use `localStorage` or `sessionStorage` to persist the team; a global React `Context` around the root `layout.tsx` is sufficient.

1. On the dynamic routes for each Pokémon, implement a mechanism to add and delete Pokémon from the team.
2. Create a floating navbar anywhere on the page which lists the current Pokémon in the team on all pages.

Error handling

You may assume that the API will never fail on valid requests. In other words, you do not have to validate the API response, however you are welcome to use Zod for additional type safety.

You may also assume that the user will never visit an invalid page.

Notes

We will be assessing functionality heavily over style, so do ensure correctness before aesthetics!