Pokemon Team Builder API Documentation

# What the site does and its purpose

To allow Pokemon fans to build their dream teams and save them on to the server.

The app features 5 pages plus a 404 page. The 5 pages are: home, create-team, create-random, view-team, and admin.

It provides two GET endpoints that are publicly accessible /getTeam?user=x and /getTeams

There are also two POST endpoints for modifying data on the server /addTeam and /deleteUser

# What part of the app does the API handle?

The /getTeam?user=x endpoint of the API is used by the view/edit page of the app and allows users to input a team saved under their username.

The /getTeams endpoint is used by the admin page and allows administrators to easily view all user data that has been saved to the server

The /addTeam endpoint is used by both the create-team and create-random pages and allows users to add and edit their teams to the server.

The /deleteUser endpoint is used by the admin page that allows admins to delete a user and their associated team from the server.

# What went right and what went wrong?

The theme came together well, I especially like the landing page. and I can see me and my friends using this App for when we play Pokemon together.

Due to sheer size of the projected DRY may’ve been violated for parts of it, but I did my best to cleanup given time constraints.

I tried to use Pure CSS as a framework instead of using Bootstrap, it was fun to work and has it’s uses especially with forms. However, I found its grids to be quite limited and found far more success with the built-in CSS Grid. I am quite thankful for this since I used to rely on Flexbox and Bootstrap grids way more than CSS Grid. I look forward to using more Grid in future projects. I also hope to try even more frameworks in the future like Foundation and Tailwind.

API was rather difficult to work with and I don’t like the fact that I must make 800+ requests to the server. Also, the API has a lot of unnecessary data for each call. I try to circumvent this by storing only the necessary Pokemon data in session storage making subsequent reloads a lot less demanding. Also, some things I wanted to do like combing fully evolved Pokemon and types was very hard to do without doing excessive API calls.

# How would you improve the app?

I would expand the landing page theme to the other pages.

A big stretch goal that I had in mind that I really want to do on my own time is to add a type effectiveness chart and give you a suggestion of the best type to add your team to help with coverage.

# How did you go above and beyond?

Used the Pure CSS framework to help build a sleek UI.

Used an external API to work with data.

Tried my best to make all page’s work for a variety of page sizes, still WIP for create-team page.

Made a media loader that supports both PNG and JPG.

Utilized sessionStorage API to help reduce external requests and improve responsiveness.

Added a JS loader on the server-side and a client-side JS file to help enforce DRY code.

Created 5 pages instead of the required 4.

Uses 4 separate 4 instead of the required 1.

Uses 3 images: 2 PNGs and 1 JPG instead of the required 1.

2 POST endpoints instead of the required 1.