**The Pokédex Flask App**

**Introduction**

The Pokédex project is a small web app for browsing the pokédex – a collective term for all Pokémon from the video game and cartoon series of the same name. The web app serves as a tool to check specific information about these Pokémon and moves, and perhaps as a learning tool to refresh ones knowledge of the poke world.

This exact pokédex version reflects the contents of the first generation of the games and contains 151 creatures, as well as their details, and 12 pokémon types. It also serves as a collection of moves/attacks these creatures are able to learn. There are 165 of these, and similarly to Pokémon, these are too divided between the 12 types and additionally, each have one of three categories assigned.

While creating this application, originality and great user experience were kept in mind. Our goal was to create a web app that is straightforward to use, looks aesthetically pleasing, contains accurate and easily readable information, and contains nostalgic elements from the Pokémon games and TV series. For simplicity, only the first generation (151 as opposed to the current 802 creatures as of 2018) was implemented.

The whole website resembles the actual pocket device for identifying the monsters (see figure XXX – compared screenshot vs TV screenshot). The screen of this device serves as the window for displaying content.

**Design**

**User Interface**

As has been previously mentioned, the central design element is the window frame in the form of a simplified Pokédex device from the games. In addition to this, a blurred background of an outside setting has been added to give the feeling of using the device in nature – to catch Pokémon. The bright hues of the green and the saturated red of the frame are very resembling of the colours used in the games. The whole frame is more or less responsive, to the extent of responding to the window width, generally speaking however is a very static element within the dynamic environment.

Despite this fact, it took extensive planning and sketching to reach the desired effect – see figures XX to YY.

Regardless of the possibility to use CSS libraries for the project, one of the main goals was to create an original experience. While bootstrap is highly customizable, our challenge was set to create an environment using plain CSS, HTML, and javascript, and to achieve details and nostalgic design elements.

In this context, any feature that reminds one of the Pokémon franchise could be considered nostalgic. Besides the obvious red frame with buttons and diodes, a few other less obvious elements were introduced. For example, within a pokémon’s page (see figure XXX), the evolutions section indicates the number of evolutions and a direct link to the page of the evolution. However, the linked pokemon’s sprites have been desaturated and darkened so as to resemble the “Who’s that Pokémon?” section of the TV series, where the viewer is prompted to guess the name of the monster. Similarly, we can only see the silhouette of the pokémon before following the link.

There are many other similar visuals, some of the more notable ones being the poke-center symbol used as the link to the index page (again, created using plain CSS), sprites taken from the games, and type labels colored the same way they are in the games.

Another main functionality of the web app is the ability to view the whole pokedex and to filter it down based on pokemon attributes (see figures XXX XXX). Along with the function of generating a random pokemon further support the learning aspect of the app and give the user a better understanding of the way the pokedex is structured.

ALL OTHER IMAGES – COPYRIGHT INFO

**Routing and URL’s**

Coming up with the API’s routing hierarchy was very dependent on the structure of the data. At first, only the Pokémon were implemented. Each of them has several attributes that are at the same level. These include pokédex number that serves as a primary key, and type (most importantly), then an alias name, measurements, description and a list of learnable moves. This lead to the basic routes “/pokemon” that branches out to “/pokemon/name”, “/pokemon/types”, and “/pokemon/types/ type”

The structure for moves takes on a similar structure, as each move also has its number, name, type, as well as a category. Regardless, there are some differences due to the fact some information is less important. Since we are displaying the type of each move within the moves table in the the “/moves” route ( see figure XX) it would be abundant information to include a “/moves/types” route. The fact that move types are identical to pokemon types also supports this conclusion. Therefore, the branch within the hieararchy has been ‘skipped’ and only the specific “moves/types/type” was implemented. Same applies for the move categories. See figure XXX

See figure XXX for a table of all routes with a short description.

For this stage of the assignment, only GET routes have been implemented, since we’re mainly retrieving data. In order to implement functionality of adding a new Pokémon or move to the dataset we would develop support for PUT and POST methods. For a detailed table of route and method combinations see figure XXX.

Besides building a base routing hierarchy, several redirects have been developed in order to make navigation through the URLs more efficient. As there are several interchangeable terms in the pokemon franchise, an effort has been put into reflecting this within the routes and make all synonymous words usable. For instance, the collective name for all pokemon is pokedex, as opposed to pokemons (which is also grammatically incorrect), hence the route “/pokedex”. However, the route for a single pokemon is ”/pokemon/name”. Considering the pokedex number of a pokemon and the name are both primary keys, they can both be used to refer to a specific creature. Therefore, all of the following combinations redirect to the specific /pokemon/name route:

* /pokemon/dex\_number
* /pokedex/name
* /pokedex/dex\_number

This way we ensure routes are very interconnected and prevent 404 status code errors

**Error handling**

Some basic error handling has been developed in the form of custom templates viewing the error code within the webpage alongside a custom apologetic message. See figure XXX.

**Data Storage**

One of the very initial stages of development was the planning of necessary data structures and their inner format as well.

Since the scope of the whole data is in the hundreds and the website does not have user support, storing in json was an easy pick. A more important decision was how this data should be organized in order to remove as many duplicates as possible and to optimize it to a certain extent. For this reason, a decision has been made to create several json files, where each mimics a relation (from relational database theory) – hence the pokemon, moves, and types files. Within each file the data is stored as an array of objects with their own primary keys (see example on figure XX).

To connect this data to the flask app, we have created a class called PokeData inside a dataconnection.py file, that we then imported and instantiated within the pokedex.py file. The PokeData class contains all functions for processing the data and returns only the necessary data in the form of dictionaries or arrays of thereof. For an example see figure XXX.

This way, we enforce separation of concerns, and make the source code cleaner in overall.

**Hypertext**

-mention linking between pokemons

-how the hierarchies link

-similar templates used over the whole app (pokedexes and reduced pokedexes, move lists and reduced move lists)

**Enhancements**

There are numerous ways of improving the web app, some of which are minor changes, and others huge alterations of functionality. Here we will explore these options as possibilities for future versions of the app.

**Routing and redirects**

The routes quite clearly reflect the way our data is structured; however, some smaller additions could fill the gaps. For instance, as we mentioned before the routes for “/moves/types” and moves/categories are missing since this is abundant information. Regardless, we could use these routes to redirect to a more logical location.

**Search function**

The ability to search the data was one of the initial plans for the app, however had to be scratched due to time constraints. This way, we could interact with our pokemon information in a whole new way.

**User support**

For user support, we would most probably need to change our data storage and introduce a database with sufficient hashing. However, having the ability to recognise distinct users would introduce a plethora of new functionality. For instance, saving favourite pokemon, a tool for building one’s teams of pokemon and comparing pokemon, a pet pokemon mini-game, a chat for interacting with others could all form the purpose and usefulness of the web app. Implementing all these features would be outside the scope of the module, but are a possibility for further direction.

**Security**

Strongly tied to the user accounts - security is a big concern for a web app. Since our website only really supports GET requests no measures were taken for securing our flask app. However, for any PUT/POST/DELETE etc. requests, we would need to make sure to take action against code injections and similar attacks.

One of the current vulnerabilities at this current stage of the project is the filter function for the pokedex. This was done by storing the arguments inside a dictionary which is then passed along to the PokeData object to retrieve a filtered pokedex in form of an array. Since we are not protecting the passed in data, this could potentially pose as a security threat. Instead, the session library should be used to create a cookie with the filter attributes.

Expand data for all generations

-how you would visually do this and how this would change the hierarchy, what changes between generations

**Design enhancements**

Since the pokedex frame already serves as a very dominant design element, it would be great to functionalise it. The fake buttons on the device could serve as an fun alternative way of navigating throught the web app.

Also, despite the frame being a little responsive, there is a lot of space for improvement. For mobile devices for instance, the whole pokedex device should be replaced with only the inner white screen bezel, as was originally planned (see figure XX).

**Error handling**

Some amount of error handling has been implemented. For instance, 404 error codes have been considered and on top of having a custom Jinja templates that seamlessly fit within the design, each route is designed to abort with a 404 error when necessary – see figure XX.

However, other error codes do not have similar handling as of the current state of the project.

**Critical Evaluation:**

+ pokedex filter and pokedex page in general + dots in pokedex - elaborate

+ who’s that pokemon

+ use of same templates over the whole design – concise and makes sense

+ clever routing redirections

+ original and timeless design

+ concise Design

-not very clear distinction between pages for moves

-deeper hierarchy

-

**Personal Evaluation**

Biggest takeaways – habit of git – space for improvement- more atomic commits

Workflow – locally – git – remote server

Obstacles:

Wrong data structure used – don’t overthink it should have gone with an array in the first place

Changes to hierarchy – routing logic

CSS challenge – nailed it – improve css knowledge

Templates for pokedex and templating in general - new kind of templating approach learned

Filter function – tasks like this make you understand your dat better

**Conclusion**

https://veekun.com/dex/downloads

sprites