Gen 4 Pokémon Database Cloud App

Levin Neumann
lwneumann@willamette.edu
github.com/lwneumann/DexApp

1 Abstract

Any fan of the Pokémon video game series who has wanted to know where to find a certain Pokémon, item, or technical machine in their game has turned to the internet and found multiple pages each with their own structure and information. After years of using these resources, I have come to be disappointed in a few features they have. This is the motivation of this project. This paper will summarize the process of planning, designing the cloud infrastructure, and creating the website.

1.1 Project Overview

There are several components this project seeks to implement. The large caveat to be mentioned is that this project will solely focus on the fourth Generation games, Heartgold and Soulsilver. These are the games I am most familiar with and so not only will be able to verify and gather information more successfully than if this was covering the current 9th generation games, but also would have around double the amount of work to do given the large increase in the content of the games given the cumulative nature of Pokémon games.

1.1.1 Pokédex entries

There are 493 Pokémon in the 4th generation games. For each one there is lots of information such as:

Name, Japanese Name, Number, Six different stats, Possible stat spreads based on nature and level, Encounter locations, Moves , Level up, Egg, Tutored, HM and TM, Base happiness, Height, Weight, Color, Classification, Egg group, Gender ratio, Effort values gained, Held items, Experience growth rate, Safari zone flee rate, ...

Each Pokémon should have their own page with all this information in a readable format, ideally with an easy way to jump between pages.

1.1.2 Events

There are many events that happen across the week based on the day. There is lot of different types of events;

1. Daily events
   (a) Events like the Goldenrod Radio Lottery that are the same each day.
   (b) Events like the Goldenrod Department Store Lottery that happen every day but has different prizes based on the day of the week.
   (c) Non Player Characters like the Weekday Siblings who appear every day but are in a different location around the map.
   (d) Non Player Characters like Gym Leaders who can be called during certain times of certain days (once you have gotten their phone number).

   ⋮

2. Weekly events like the Bug Catching Contest which happens Tuesdays, Thursdays, and Saturdays.

⋮

Although most of these will be static interactions, there will need to be a dynamic display of the events happening today[1]. These events can be placed into one page, but will need to have some elements that change day to day.

1.1.3 Additional Pages

There is a lot of information that could be added that likely will not be able to fit in the scope of the project. That being said some pages that will be added are a page with information on moves, and a page with information on abilities.

1.1.4 PDF

For several years I have been writing a document that seeks to cover information on Pokémon Heartgold and Soulsilver. This does not cover things such as the Pokédex entries mentioned in 1.1.1 so it would not redundant to include it as well as it has a large amount of other relevant game information that likely would not be able to be given their own respective pages on this website given the time and scope of this project. The page including this would be very simple and just display the PDF. Additionally this would be easy to update as the PDF gets new versions as the file in the cloud application can just be updated.

2 Cloud App Infrastructure

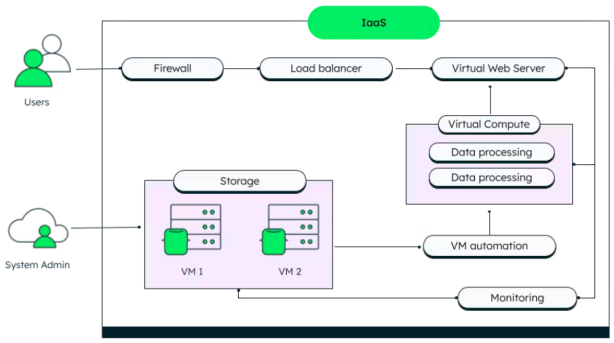
When creating a cloud application there are many choices to be made. With the context given in 1.1 we are now able to discuss which infrastructure would make the most sense to host this application. There are many options we can choose for each section. For all of them, first they will be broken down, then the choice for the best option for this project will be discussed.

[1] meaning whatever day the website is being viewed.

2.1 Service Model

2.1.1 Infrastructure as a Service (IaaS)

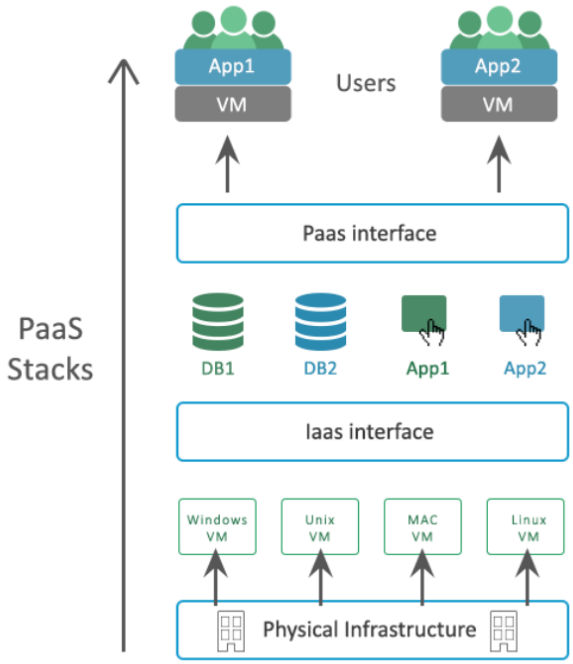
This is useful as it allows for much more control from the customer given that they are more or less buying simply space. Because of this there is high performance without any (basically) bloatware from the provider. Here from a MongoDB[11] diagram for IaaS.



However the downside to this is that because of the reasons mentioned before you have to spend much more time to set up the space in the way you want, which can often cause problems due to the complexity. Additionally the upfront costs associated with these can be problematic if you were intending to use monetization to fund your application.

2.1.2 Platform as a Service (PaaS)

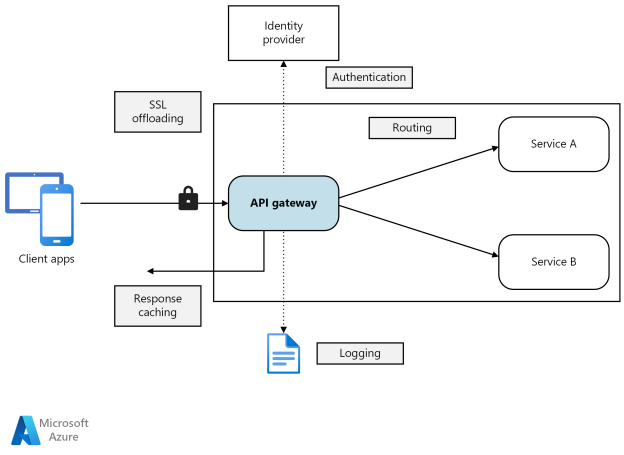
Using Platform as a Service is useful as you are able to drop your front end onto what the providers use which enables the customer to not worry about handling middleware and back end. This makes it a lot more accessible without having to get an expert in these fields and just focus on front end. You are able to pick the language of middleware and back end as well giving more control. Here is a MilesWeb[10] diagram showing the structure of a Platform as a Service.



The downside however is given the vendor is providing lots of service to the user, it means that there is limited control to the user as well as can often lead to vendor lock in as you need to replicate whatever structure the provider was using to create their infrastructure in order to move out to another service.

2.1.3 HTTP Web API Gateway

HTTP Web API Gateways are a very simple implementation as for development. You don't need to work very hard to develop this meaning the development process is easier. This offers increased security over other options as you have much more limited access from users. This often is much more scalable as again it is much more simple than other options. Here is a diagram from Azure[1] showing the structure of most API gateways.



This brings limited control as the consumer will not be able to provide a lot of the structure as the vast majority is given by the provider. Additionally this model may end up costing more for you depending on implementation.

2.1.4 Conclusion

This project will use Platform as a Service. This ideal for hosting Flask apps, managing infrastructure with a focus on development. Since there is Python support, automatic scaling, and built-in monitoring, this ensures high performance, security, and easy maintenance, making it a cost-effective choice especially for the scale of the project.

2.2 Database

My database is comprised from data scraped from 494 websites combined with another CSV file. It is all very structured as it is Pokémon IDs, their locations, moves, and so on. The data itself will never need to be scaled as they will not go back and add new data to a 14+ year old game. In order to maximize the efficiency I plan to compile all HTML pages with the information before publishing to optimize the user experience. If there was a Database it would be a relational database however given that I am using Flask, it would not make a lot of sense to place it into a relational database even though I have a well-defined structure.

There is a pdf with over 80 pages and lots of information but it is largely text and images so will remain in this format just to be read by the user.

There is a small handful of other pieces of information such as weekly and daily events but these will remain in python as they are all very small so make no sense to be stored outside of the python file.

Therefore I will use a file structure with static pages for my project rather than a database.

2.3 API Protocols

2.3.1 REST

REST is ideal for stateless, resource-based operations such as CRUD actions, making it an excellent choice for APIs that need to handle standard data operations without requiring ongoing connections. Its main strengths lie in its simplicity, broad compatibility, and widespread browser support, making it both easy to implement and highly accessible. However, REST can suffer from higher latency compared to some newer protocols and is not optimized for real-time communication needs, which limits its effectiveness in applications where instantaneous feedback is necessary.

2.3.2 SOAP

SOAP is a strong fit for enterprise-level, security-focused applications that need robust, transactional reliability. It offers built-in security through WS-Security and adheres to strict standards, making it highly suitable for distributed environments where data integrity and security are critical. On the downside, SOAP is significantly more complex and heavier than REST, resulting in more demanding implementations and generally slower performance, which may be a consideration for less resource-intensive applications.

2.3.3 gRPC

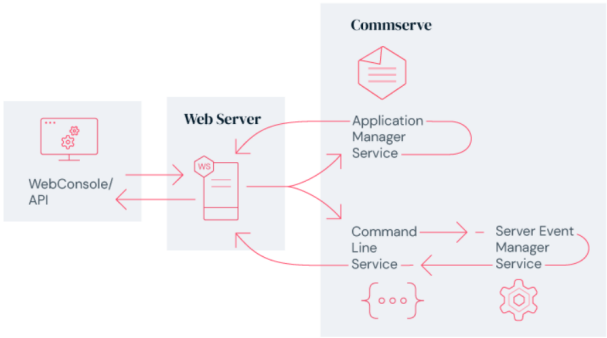
gRPC excels in scenarios that require high performance and low latency, particularly within microservices ecosystems. It supports bi-directional streaming and uses Protocol Buffers for efficient data serialization, providing both speed and low latency. However, gRPCs setup can be more complex, and it requires dedicated libraries for each programming language, adding to the initial implementation complexity and necessitating specific expertise in its use.

2.3.4 WebSocket

WebSocket is designed for real-time, bi-directional communication, making it a popular choice for applications like chat platforms and live data feeds. It offers low latency and full-duplex communication, which means that data can flow freely between client and server without the need for multiple requests. However, WebSocket is unsuitable for stateless operations, as it requires maintaining an open connection, which can limit its scalability in certain environments.

2.3.5 Webhooks

Webhooks are highly effective for event-driven communication, ideal for sending notifications or automating workflows based on specific triggers. They are simple to set up for one-way, event-based workflows and are a lightweight solution when real-time, active communication is not necessary. However, Webhooks are limited by their unidirectional nature and are dependent on external services to initiate requests, which may limit their flexibility in some applications. Here is a diagram from Commvault[2].



2.3.6 Conclusion

This project will go with REST. Since nearly all pages are static, they wont need real-time or event-driven communication. Additionally, REST is highly compatible with Flask and allows efficient, stateless handling of HTTP requests to serve pages or data as needed. RESTs simplicity and Flasks lightweight nature make them ideal for delivering static content without requiring complex communication protocols.

Flask does have integration with other options such as Web sockets using extensions, so if the nature of the site changes in the future, it will not be locked into REST.

3 Pokémon Data

Now that the context and structure of the project is decided, the main work of this project can be covered. Actually populating the website and their pages requires lots of information. As mentioned in 1.1.1, there are 493 Pokémon each with a significant amount of unique data that needs to be gathered, processed, then displayed.

3.1 Data Gathering

The majority of the data gathered using web scraping. There are a handful pieces of missing information that were filled using various csvs and my own knowledge from the game.

3.1.1 Pokédex Entries

There are many websites that already have a significant amount of data. The majority of the data being used in this project was web scraped off of the website Serebii[17]. The url takes the form:

serebii.net/pokedex-dp/[Pokémon Number].shtml

The Pokémon's numbers needed to be filled with 0 as well, so 001, 002, ... 058 ... 493 and so on. Using this it was then fairly easy to save all 493 pages that are needed.

3.1.2 Other

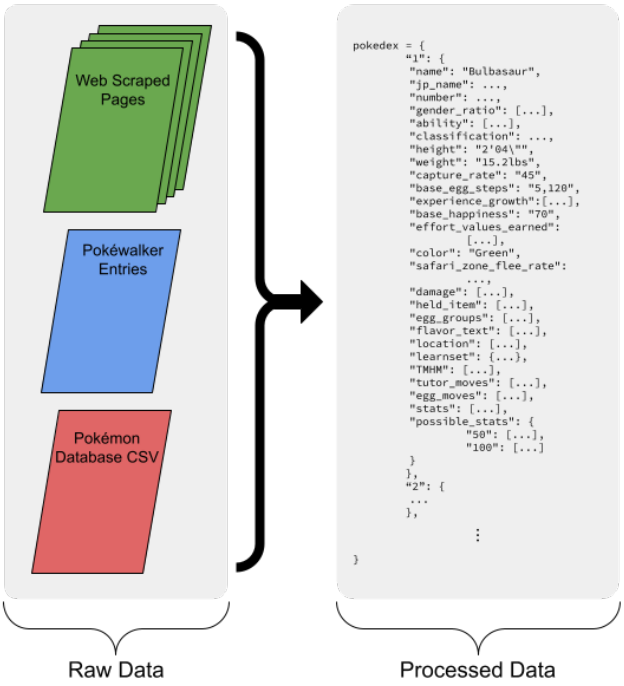
Serebii[17] has nearly all the information required for this project however there are some elements that were difficult to extract (as discussed in 3.2), as well as general information such as the descriptions of abilities, and the location of Pokémon from the Pokéwalker accessory which is unique Heartgold and Soulsilver games. Because of this several other sources were used to get a more complete pool of information to be used. These were as follows:

1. A Kaggle Database[8] that has every Pokémon a easy way to access important information about each of them such as stats, typing, what generation they're from, and if they are Legendary[2] or not.
2. Another great website Psypoke[15] has a list of all abilities from the fourth generation of games including their description, overworld effects, and which Pokémon can have them.
3. The Pokéwalker has 27 different routes, each with 6 slots of Pokémon you can encounter. There is more information within that such as fixed genders, number of steps needed[3], held items, ... Bulbagarden has Bulbapedia[5] which is a similar website to a lot of the other websites mentioned before. They have a list of Pokémon from the Pokéwalker, which was also taken for this project.

With the prior pages scraped from 3.1.1 and these, this is a complete enough amount of data to begin turning the raw data into something useful for the website.

3.2 Data Processing

The general overview of turning all the raw data into something useful. Generally it will look like this:



Where we take and combine all the various data we gathered into one unified item.

3.2.1 Reading Raw Pages

In order to actually use the data, it needed significant work. This was by far the most time intensive part of the project, taking many hours. The main issue came from the fact that the text files from Serebii[17] usually sat around 2,400 lines. Also all the important information such as moves the Pokémon learned typically were all in one line. A python file was used to go line by line through the text. It would grab easy text like "Name \n Bulbasaur", and pass off the

[2]The Pokémon world has very fleshed out lore and so there are many Pokémon that are considered Mythic and Legendary.
[3]The Pokéwalker is a pedometer with internal game play and connectivity to the game

very long lines to be decoded from helper functions designed for each specific type of information gathered this way.

Each page was turning into a dictionary for each Pokémon with 25 keys containing their unique information as mentioned in 1.1.1. When spread out to be readable these were usually around 250 lines.

After this, each of the processed files are combined into one (very large) final dictionary with a key for each Pokémon, and those each contain their own information.

3.2.2 Processing Additional Information

There is still Database[8] and the Pokéwalker[5] that need to be included. The approach was to take all usable information from these two then combine them into a final dictionary to be used for the website creation in 4.

3.2.2.1 Pokémon Database

The CSV from the Pokémon Database[8] was already a CSV so needed no additional work, just deciding what to remove then include.

The CSV includes much more modern Pokémon up to generation 6. This means that it also includes the alternate Mega forms of Pokémon which were introduced in the 6th generation as well. These all needed to be removed. So this python file worked through every entry until it passed the fourth generation. Within that search it also had to discard all Mega Pokémon, accounting for edge cases such as the Pokémon ***Mega***nium.

Now that the data has been refined to what we want to use, we can inject the desired components into our dictionary. Although there was a large amount of information extracted in the Raw Data (3.2.1) there was equally a large amount of information that was poorly formatted such stats and Pokémon typing. Additionally some information such as the Pokémon's generation and if it is Legendary or not. would be unnecessarily difficult to extract from the raw data when it is just given in the CSV. These were all then added as well.

3.2.2.2 PokéWalker

This information was not web scraped but rather simply copy and pasted from Bulbapedia[5]. Because of this a separate file was used to read through, strip, split, and read the information from the text file. Most of the information was not needed, so it gathered the Pokémon, if they had an item, and the area(s) they appear. This yielded 118 entries.

3.2.2.3 Abilities

There is a list of different Pokémon abilities on Psypoke[15], which has the ability name, the description, and the Pokémon who can have that ability. There was a few hiccups with characters encoding such as juggling turning "â ™€" to ♀, and "â ™," to ♂. Once this was done that table was turned into a dictionary with ability names, descriptions, and the Pokémon who can have that ability. This is useful for a few reasons. Individual pages can now have their abilities with a description and there can be a whole page for all the abilities.

3.2.2.4 Moves

This was an obnoxious process to go through. Initially most websites will list "Gen 4 Pokémon moves" but actually mean just the moves exclusive to the fourth generation not all moves available. Serebii[16] has a listing for all moves but are filtered by type and category meaning there is not just one page for all moves. Additionally due to formatting only a handful of the data we need was redly available, so another source - Strategy Wiki[23] - was used to get all moves with the remaining information such as types and categories. These are defining features of a move, so very necessary. There also came a problem of filtering the data to be useable. There is a list of names that were phrased differently between each source;

$$\begin{aligned}
\text{Double-edge} &\longleftrightarrow \text{Double-Edge}\\
\text{Doubleslap} &\longleftrightarrow \text{Double Slap}\\
\text{Thunderpunch} &\longleftrightarrow \text{Thunder Punch}\\
\text{Twineedle} &\longleftrightarrow \text{Twin Needle}\\
\text{Vicegrip} &\longleftrightarrow \text{Vice Grip}\\
\text{Mud-slap} &\longleftrightarrow \text{Mud-Slap}\\
\text{Roar Of Time} &\longleftrightarrow \text{Roar of Time}\\
\text{Sonicboom} &\longleftrightarrow \text{SonicBoom}\\
\text{Conversion 2} &\longleftrightarrow \text{Conversion2}\\
\text{Lock-on} &\longleftrightarrow \text{Lock-On}\\
\text{Poisonpowder} &\longleftrightarrow \text{Poison Powder}\\
\text{Sand-attack} &\longleftrightarrow \text{Sand-Attack}\\
\text{Will-o-wisp} &\longleftrightarrow \text{Will-O-Wisp}\\
&\vdots
\end{aligned}$$

Finding all the differences and ensuring that each time a name from one or the other is being used, it is correctly adapted to the right side. There were a lot of key errors before these were all tracked down. There was again the dance of character encoding and "Ã©" had to be changed to é.

There was still additional information missing such as the item that can teach the moves - TMs and HMs - as well as the separate numbering of the moves. Bulbapedia[6] has a list of TMs and HMs as well as Bulbapedia's other list[4] of moves with numbers. The actual move information isn't really used from the later aside form the number of the move and the generation they are from.

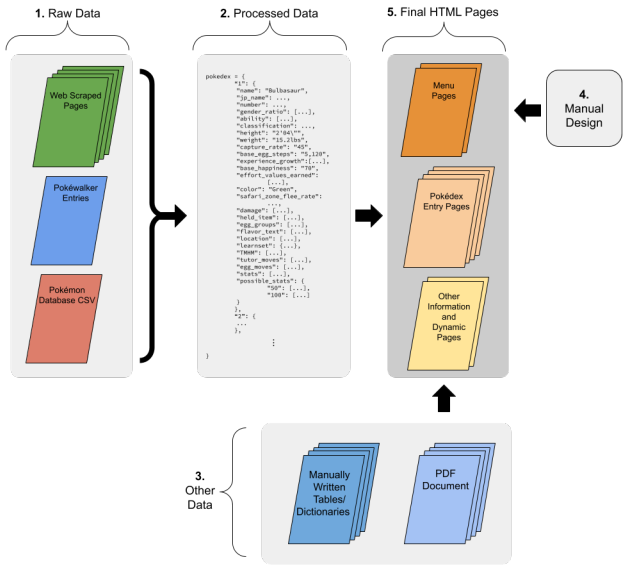
Using a similar method that was mentioned before, these were all combined into one dictionary with the final move information.

3.2.3 Compiling Additional Information

There is all the additional information gathered in 3.1.2 currently not included in the new dictionary generated from 3.2.1, so we need to include the Pokémon Database CSV[8], and the information about the Pokéwalker[5]. This was another python file which read and initiated the old dictionary, then went through each entry (a Pokémon), and added the information from the CSV. Additionally it checked if the Pokémon had any information from the Pokéwalker's[5] information. Once that was complete the data had been fully compiled, it was time to create the website.

4 Website

The general website will be structured as shown below. As mentioned in 2.2, there is no database needed for this project and static web pages will be used as the information has no need to be dynamic. Although most of the information is in good shape at this point, it still needs to be moved into a cohesive structure, and worst of all - made to look nice. In short all that is left is to coerce everything into the final form; a complete website. We can now update the graphic in 3.2 to show the complete flow of the project.



4.1 File Structure

Flask allows for template files. This was used to ensure that everything looked the same. Every page starts with base.html which provides the background, buttons, and so on. All Pokédex pages also went through 000.html, which has all the formatting, pictures, tables, ... to show entry information.

4.2 Resources

There was additional work done aside from information gathering that was performed in order to get button images, sprites, other sprites, and so on.

4.2.1 Images

Many of these additional images came from all sorts of various sources.

4.2.1.1 Emulator

In order to get certain pictures for button icons, a DeSmeMe emulator[3] was used to get screenshots in order to use as much in game imagery as possible such as the player's home as seen above.

4.2.1.2 Sprites

Each Pokémon has many variations and versions of themselves. There are icons, sprites, shiny sprites, female sprites, female shiny sprites, ... Pokencyclopedia has a list of all battle sprites[14] which are displayed on each Pokédex page and all animates sprites[13] which are used as navigation buttons on the bottom of each Pokédex page.

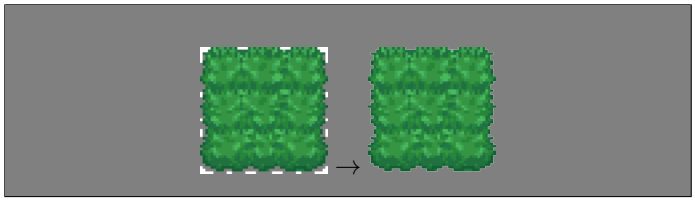
4.2.1.3 Other Images

There are a handful of other images such as move icons found on Serebii's attack page[16].

In order to get some of the bushes and trees for the background there was an old thread on The VG Resources[24] that had the outdoor tiles.

4.2.2 Image Processing

Most of the images had backgrounds which can be seen against the grey below.



This was solved fairly easily with a quick manual removal of the undesirable pixels using python. All type icons, and move categories were also fixed this way to be presentable for the website.

4.3 Events

Every day in the games there are a large amount of unique events happening based on what day it is. These are all covered in the PDF in 1.1.4. These were manually written

into a dynamic page which will reflect not only all events but also the events of the current day.

4.4 Pokédex Entries

Once the 000.html template from 4.1 had been fully constructed with all the edge cases such as if the Pokémon has a Pokéwalker course, if there are items on the Pokéwalker courses, if there are female sprites, if the Pokémon learns any TM HM, egg moves, or tutor moves, and so on.

Once this was done, yet another python file was used to collect, filter, and fill the template for all 493 Pokémon in order to inject their information into the 000.html template. There were a lot of changes to make:

1. All items had to be strings.
2. Several needed to surrounded by quotes.
3. Each stats has a maximum and minimum gathered from levels 50 and 100.
4. Japanese names needed to have the romaji serrated from the hiragana.
5. Boolens for all edge cases

⋮

4.5 Error Pages

Each error page has a custom message written to accompany the error, sharing the common background and menu options. This was made using the sprites for the Unown Pokémon which has variants for each letter and some punctuation.

5 Conclusion

5.1 Effective Choices

The use of static pages instead of databases is a good choice for the project allowing for faster load times and less processing as the project doesn't need to query and populate for each page. This is very useful as users will likely skip around from page to page meaning that the faster this is done the better.

I am coming from no experience of web design or HTML. On that front using Flask[7] to control the website was easy to learn and implement static pages, inheritance, populating pages with data, for loops, ... was a good choice. Natrually given someone with more experience this could vary.

5.2 Project Limitations

5.2.1 Pokémon Data

In it's current state, there are several important pieces missing from the data. This comes from a choice to triage the scale of the dataset in order to complete the project within the given scale and time frame. Briefly here is some of the missing data.

Pokémon evolution is one of the largest components of the game, and as is, it is not reflected at all within the project. On the surface you may think that this is just including a chain such as Bulbasaur →Ivysaur →Venosaur. The issue is that many Pokémon have multiple different criteria and even multiple different Pokémon they can evolve into based on their happiness, items, trading, gender, and so on.

There are many Pokémon that have multiple forms that are currently ignored. Generally for Pokémon like Wormdam this isn't really an issue as frankly they see no play[21]. Unfortunately there are Pokémon such as Deoxys[18], Giratina[19], Shaymin-Sky[20] ...which are ranked in the Ubers[4] tier of fourth generation competitive play.

This is an issue both for casual and competitive players. Notably this is not an issue with the choices made for the structure of the project but just a time limitation as there are 493 Pokémon each with unique relations and states that need to be manually addressed.

5.2.2 Additional Features

There are a very large number of features, tools, and information that can be added. For example a team weakness calculator[9] or a team builder[12] with more analysis of coverage, weakness, susceptibility to Attack or Special Attacks, ...

Many new pages could be added such as all items, more information on moves such as who and how Pokémon can learn them, specific information such as story walk through, RNG manipulation[22] ...

5.3 Conclusion

The majority of the drawbacks lie in the data gathered and the vast unique nature of Pokémon games. Projects like this are generally fairly sparse when it comes to features and remarkably rich in content and information that can be shared. Given more time the usability and scope of this project would greatly increase.

[4] This is the most powerful tier of competition, which no bans on Pokémon whereas the lower tiers all ban the Pokémon from tiers above allowing for a way to play with a variety of power level.

6 References

[1] API gateways - Azure Architecture Center, learn.microsoft.com/en-us/azure/architecture/microservices/design/gateway

[2] REST API Flow, documentation.commvault.com/2024/essential/rest_api_flow.html

[3] DeSmuMe Download, desmume.com/download.html

[4] List of moves, bulbapedia.bulbagarden.net/wiki/List_of_moves

[5] List of Pokémon found through the Pokéwalker, bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_found_through_the_Pok%C3%A9walker

[6] List of TM and HM locations in Generation IV, bulbapedia.bulbagarden.net/wiki/List_of_TM_and_HM_locations_in_Generation_IV

[7] Welcome to Flask - Flask Documentation (3.1.x), flask.palletsprojects.com/en/stable/

[8] Alberto Barradas, Pokemon with stats, kaggle.com/datasets/abcsds/pokemon

[9] Marriland Team Builder for Pokémon Teams, marriland.com/tools/team-builder/en/

[10] What is Paas (Platform as a Service) in Cloud Computing?, milesweb.com/hosting/cloud-hosting/what-is-paas

[11] What is Iaas? Infastructure As A Service, mongodb.com/resources/basics/cloud-explained/iaas-infrastructure-as-a-service

[12] My Pokemon Team, mypokemonteam.com

[13] SpriteDex: Animated Icons Generation IV, pokencyclopedia.info/en/index.php?id=sprites/menu-icons/ico-a_old

[14] SpriteDex: Generation IV, pokencyclopedia.info/en/index.php?id=sprites/gen4

[15] Pokemon Abilities List (Generation IV), psypokes.com/dpphgss/abilities.php

[16] DP Attackdex, serebii.net/attackdex-dp/

[17] #001 Bulbasaur, serebii.net/pokedex-dp/001.shtml

[18] Deoxys — DP — Smohon Stategy Pokedex, smogon.com/dex/dp/pokemon/deoxys/

[19] Giratina — DP — Smohon Stategy Pokedex, smogon.com/dex/dp/pokemon/giratina/

[20] Shaymin-Sky — DP — Smohon Stategy Pokedex, smogon.com/dex/dp/pokemon/shaymin-sky/

[21] Wormadam — DP — Smogon Strategy Pokedex, smogon.com/dex/dp/pokemon/wormadam/

[22] The DPP / HGSS RNG Manipulation Guide - Smogon Univeristy, smogon.com/ingame/rng/dpphgss_rng_intro

[23] Pokémon/List of moves, strategywiki.org/wiki/Pok%C3%A9mon/List_of_Moves

[24] Pokemon Heart Gold / Soul Silver Tilesets, vg-resource.com/thread-12546.html