Togepedia
Morning-10
Phase III

Project Lead: Jaime Tan (jt39777) jaime.j.tan@gmail.com JaimeTan
Tristan McDaniel(thm443) t.mcdaniel@utexas.edu tmcdaniel511
Jimmy Phan (jp54694) Jimmyphan@utexas.edu JP1204
Mihir Shah (mss4367) mshah99@utexas.edu CodiWanKenobi

GitHub Repository
Togepedia

04/24/2020

Motivation

We wanted to create a one-stop-shop for Pokémon data. We provide information on Pokémon stats (Attack, Defense, Special Attack, Special Defense, and Speed), locations where certain Pokémon can be found (Routes, caves, different methods, etc.), types, movesets (possible moves that each Pokemon can learn), abilities, and a team builder to optimize your Pokémon team (users can mix and match different Pokemon with a wide array of different moves). With each pokemon having different moves and abilities, we can help link these different attributes through each other providing a way for the user to connect a pokemon, a move, and an ability with the ease of a click on each informational page.

Users

Users of our site range from casual Pokémon video game players who merely want to play through the story mode of the game and need minimal information about Pokemon, moves, abilities, items, etc. to the cream of the crop competitive Pokémon players who invest more time into the post-game content of Pokemon series games where they build the best Pokemon squads imaginable. We wanted to provide enough information so that everyone can add to their base of Pokémon knowledge regardless of your skill level and dedication to the games. We have listed below the different users that will interact with our website.

User Stories

As a person who has carpal tunnel syndrome, I want to be able to have X number of things on a page (e.g. Pokémon, items, moves, abilities, etc.) so that I do not have to continuously scroll.
[Estimate: Phase II; Completed: Phase II (7 hours)]

As a programmer, I would like to be able to use the data from Togepedia so that I can make an extension of this IMDB-style project.
[Estimate: Phase II; Completed: Phase II (15 hours)]

As a casual Pokémon gamer, I would like to see what types are super effective, not very effective, and do not affect other types because I do not have the memory to be able to remember all of the different interactions.
[Estimate: Phase I; Completed: Phase I (5 hours)]

As any user of Togepedia, I would like a navbar that allows other users and me to visit other parts of the site (i.e. models, extra data, feedback, login, etc.) in an intuitive fashion so that I do not have to learn a confusing layout to access parts of the site.
[Estimate: Phase I; Completed: Phase I (6 hours)]

As a Pokémon enthusiast, I want to be able to see where I can find different Pokémon so that I can catch ‘em all.
[Estimate: Phase I; Completed: -----]

As a Pokémon enthusiast, I would like to create my very own team to see how compatible they would be instead of wasting time in the game collecting them.
[Estimate: Phase II; Completed: 10 hours]

As a competitive Pokémon player, I would like to see which Pokémon would be suited for sweeper, wall, staller, phazer/hazer, etc. roles based on their stats and moves that they can learn so that I can build an optimal team for the competitive environment.
[Estimate: Phase II; Completed: Phase III (14 hours)]

As an avid player of the Pokemon series, I want to see all the in-game items I can obtain so that I can better prepare myself for beating the game and for more insight into post-game content.
[Estimate: Phase II; Completed: Phase II 4 hours]

As a casual Pokémon gamer, I would like to learn more about the different abilities that Pokémon can possess so I know what I am up against when I face the Pokémon Gym Leaders as well as the Elite Four and Champion.
[Estimate: Phase I; Completed: Phase II (4 hours)]

As a Pokémon enthusiast, I want to ensure that sites that are associated with the Pokémon brand can be improved based on user feedback so that users of the site can ensure that they get an experience out of the site that they desire.
[Estimate: Phase III; -----]

As a competitive Pokémon player/casual Pokémon gamer I would like to see all of the moves that Pokémon can learn so that I know the effects, type, and stats of each move when I am confronted with an unfamiliar move in battle.
[Estimate: Phase II; Completed: Phase II (3 hours)]

As a user of the site, I would like to see an aesthetically pleasing layout so that I am intrigued to spend a long time on the website.
[Estimate: Phase I; Completed: Phase I (8 hours)]

As a generic user of Togepedia, I would like to see an About Us page so that I can see the developers of the site and get an insight into the thought process of development.
[Estimate: Phase I; Completed: Phase I (9 hours)]

As a Pokemon enthusiast, I would like to be able to access all 807 Pokemon so that I can learn about their types, stats, abilities, etc.
[Estimate: Phase II; Completed: Phase II (4 hours)]

As someone who forgets user IDs and passwords easily, I would like to be able to login to the website using my Google account so I can easily login without remembering a unique ID for Togepedia.
[Estimate: Phase III; -----]

As someone who likes to keep things in an orderly fashion, I would like to be able to sort things by alphabetical order and also be able to sort Pokemon by their Pokedex number.
[Estimated: Phase III; Completed: Phase III (12 hours)]

As a competitive Pokemon player, I want to be able to edit the stats of my Pokemon team on the fly so that I can see potential downsides if I get a great Pokemon with bad stats or the potential upsides if I get a bad Pokemon with excellent stats.
[Estimated: Phase III; Completed: Phase III (12 hours)]

As a competitive Pokemon player, I want to be able to quickly see what Pokemon and moves are of a certain type so that I can refine my competitive Pokemon team with ease.
[Estimated: Phase III; Completed: Phase III (7 hours)]

As someone with little patience, I want to be able to quickly search for different Pokemon, moves, items, and abilities so that I do not have to look through each page to find what I am looking for.
[Estimated: Phase III; Completed: Phase III (13 hours)]

As a Pokemon player, I want to be able to filter Pokemon and moves by types so that I can quickly find all my favorite Ghost and Psychic-type Pokemon
[Estimated: Phase III; Completed: Phase III (12 hours)]

As a casual Pokemon player, I would like to see linkage between Pokemon and different moves to see what Pokemon can learn what moves and what moves can be learned by certain Pokemon.
[Estimated: Phase III; Completed: Phase III (14 hours)]

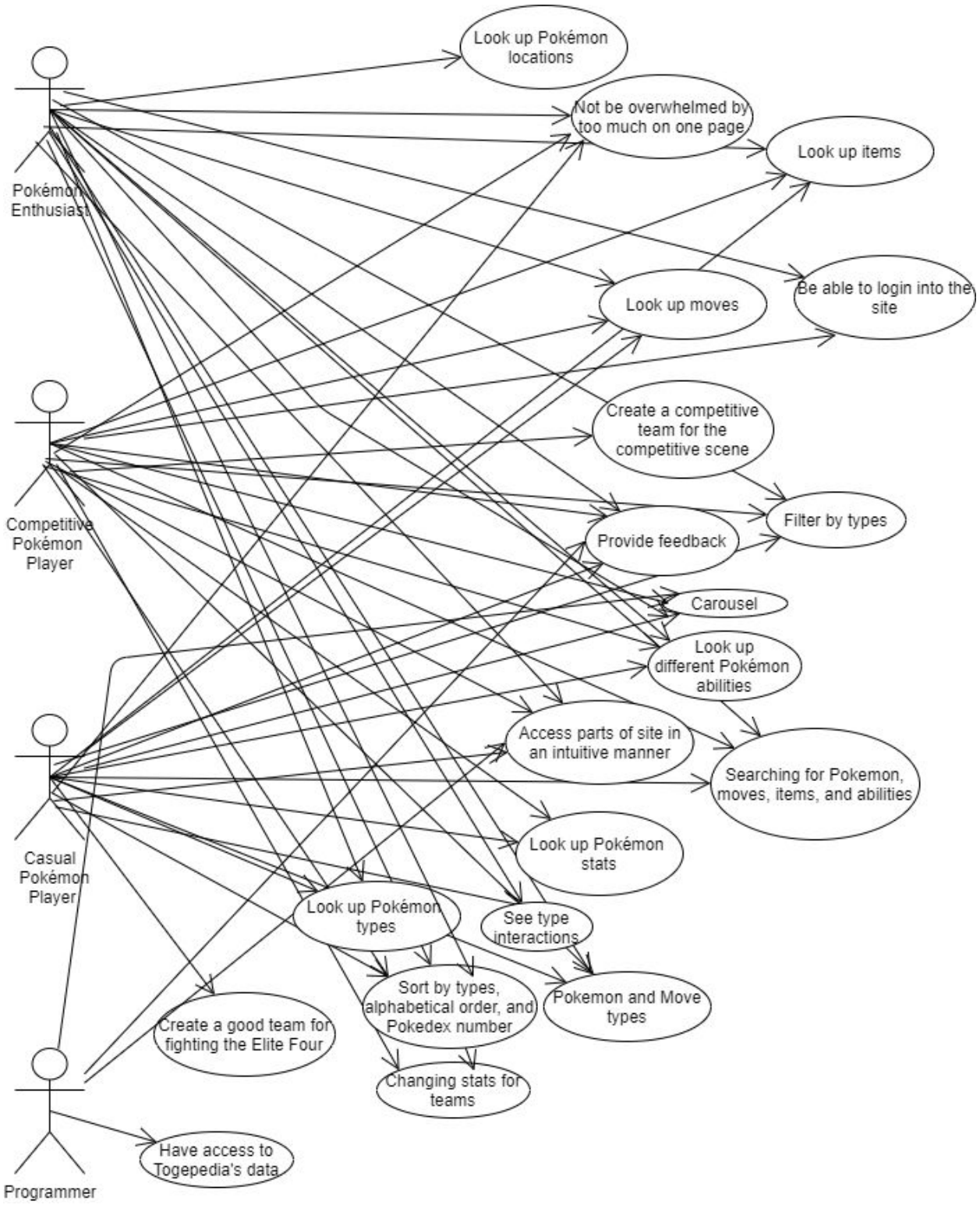
As a casual Pokemon player, I want to be able to see what Pokemon can have certain abilities and what abilities can be found on certain Pokemon so that I can see the synergy between Pokemon and abilities.
[Estimated: Phase III; Completed: Phase III (14 hours)]

As a casual Pokemon player, I would like to see linkage between moves and abilities so that I can see synergy between abilities and moves.
[Estimated: Phase III; Completed: Phase III (14 hours)]

As a competitive Pokemon player, I want to make sure I can access my team at all times regardless of the status of the underlying API for the website so that I am always on top of my game.
[Estimated: Phase III; Completed: Phase III (15 hours)]

As a competitive Pokemon player, I want to be able to see what Items boost the effectiveness of different moves so that I can optimize what items to place on different Pokemon with certain moves.
[Estimated: Phase III; -----]

Use Case Diagram



Design

When the user first enters the website, they are greeted with a carousel that showcases several Pokémon which adds to the overall aesthetic of the website. At the top of the webpage, the user has the option to go to different pages via navigation bar including the Pokedex, Moves, Abilities, Items, Types, Team Builder, About Us, and Feedback pages as well as a Signin button. Additionally, the logo in the middle of the navigation bar allows the user to go to the home page.

If the user clicks on the “Pokedex” link, they are taken to a search page for all 807 Pokémon ranging from the Kanto region to the Galar region. Basic details of each Pokémon are listed including their name, national Pokedex number, and image. Users can filter Pokemon by their up to two types and can additionally search for Pokemon by name. If the user clicks on a “Learn more” button under a particular Pokémon, they are taken to that specific Pokémon’s page which contains details about their stats, moves, weight, type, etc.

The “Moves” tab is similar to the “Pokedex” tab in the sense that it allows users to search up different Pokémon moves. Users can filter moves by different types as well as search for specific moves. Clicking a particular move’s name will send the user to a page that gives more details about the move such as a description of the move, its power, pp, accuracy, and additional effects.

The “Abilities” tab allows the user to search up Pokémon abilities.

The “Items” tab allows the user to search up different hold items that are available for use by Pokémon.

The “Type” tab lists all 18 Pokémon types as well as a chart showcasing the interactions between each type. If the user clicks on any type, we will have all Pokémon as well as moves of the type listed [WIP].

The “Team Builder” is currently a very rough skeleton that will allow users to choose a party of 6 Pokémon. We would like to have both competitive as well as casual players of the Pokémon games to be able to utilize this page for their specific needs. The page is updated to where you can select up to six pokemon and place them into a team

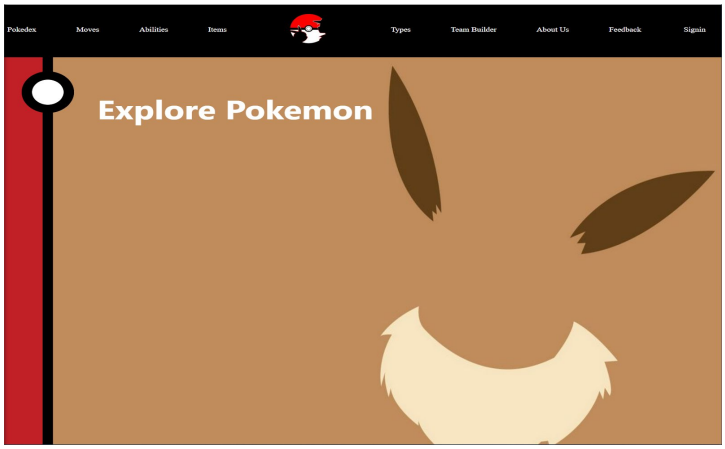
The “About Us” tab gives a short bio of each team member. It also provides a list of contributions made.

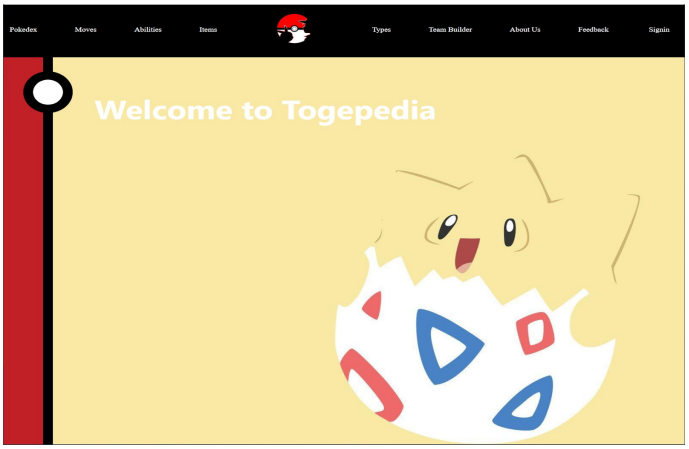
The “Feedback” tab links to a feedback form skeleton where we would allow signed-in users to provide us with feedback on our site.

The “Signin” allows the user to select a Google account through which the user will be able to save their team from the “Team Builder” page. Currently we have not fully implemented this feature. It works on localhost but does not work when we deploy it onto Google App Engine

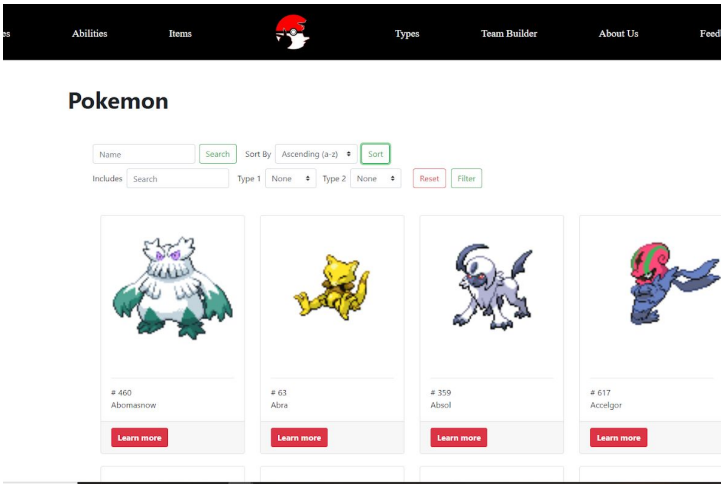
The logo in the middle of the navigation bar will lead the user back to the home page where we can once again see the carousel.

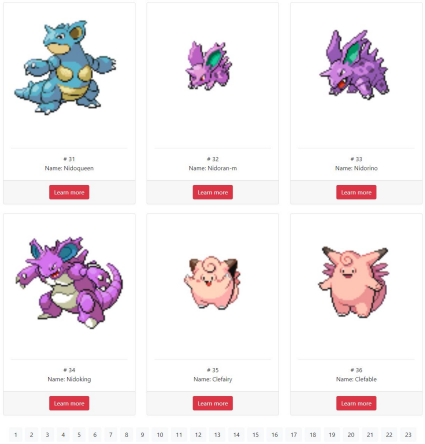
Screenshots of the Site and Data



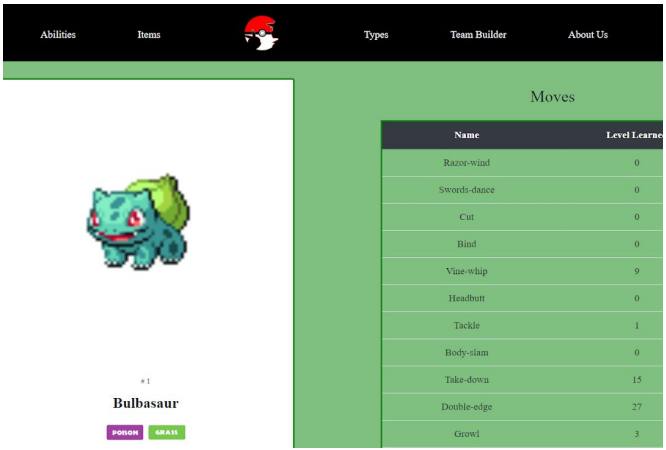


- Front page/splash page with a working carousel

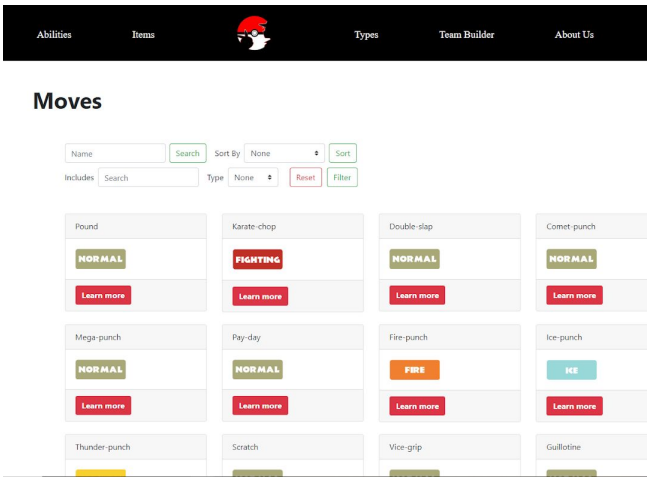


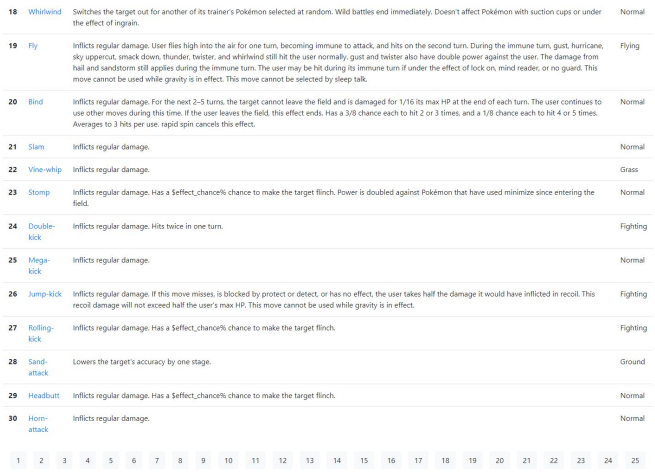


- List of Pokemon with working pagination, each shows 30 pokemon



- Pokemon Info Page - Displays our data from our database still WIP (Loading takes a bit when fetching the data)

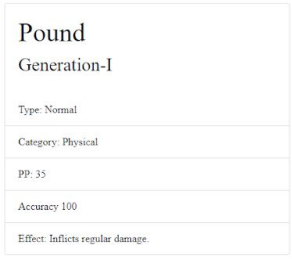


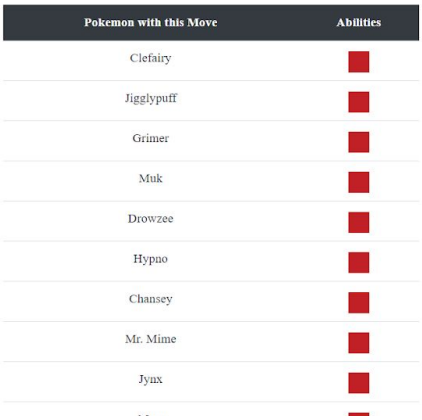


18	Whirlwind	Switches the target out for another of its trainer's Pokémon selected at random. Wild battles end immediately. Doesn't affect Pokémon with suction cups or under the effect of Ingrain.	Normal
19	Fly	Inflicts regular damage. User flies high into the air for one turn, becoming immune to attack, and hits on the second turn. During the immune turn, gust, hurricane, sky uppercut, smack down, thunder, twister, and whirlwind still hit the user normally. gust and twister also have double power against the user. The damage from hail and sandstorm still applies during the immune turn. The user may be hit during its immune turn if under the effect of lock on, mind reader, or no guard. This move cannot be used while gravity is in effect. This move cannot be selected by sleep talk.	Flying
20	Bind	Inflicts regular damage. For the next 2–5 turns, the target cannot leave the field and is damaged for 1/16 its max HP at the end of each turn. The user continues to use other moves during this time. If the user leaves the field, this effect ends. Has a 3/8 chance each to hit 2 or 3 times, and a 1/8 chance each to hit 4 or 5 times. Averages to 3 hits per use. rapid spin cancels this effect.	Normal
21	Slam	Inflicts regular damage.	Normal
22	Vine-whip	Inflicts regular damage.	Grass
23	Stomp	Inflicts regular damage. Has a $effect_chance% chance to make the target flinch. Power is doubled against Pokémon that have used minimize since entering the field.	Normal
24	Double-kick	Inflicts regular damage. Hits twice in one turn.	Fighting
25	Mega-kick	Inflicts regular damage.	Normal
26	Jump-kick	Inflicts regular damage. If this move misses, is blocked by protect or detect, or has no effect, the user takes half the damage it would have inflicted in recoil. This recoil damage will not exceed half the user's max HP. This move cannot be used while gravity is in effect.	Fighting
27	Rolling-kick	Inflicts regular damage. Has a $effect_chance% chance to make the target flinch.	Fighting
28	Sand-attack	Lowers the target's accuracy by one stage.	Ground
29	Headbutt	Inflicts regular damage. Has a $effect_chance% chance to make the target flinch.	Normal
30	Horn-attack	Inflicts regular damage.	Normal

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

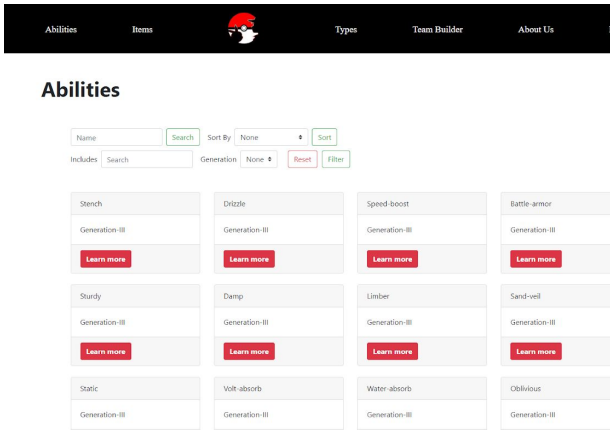
- List of Moves with working pagination, each page shows 30 moves

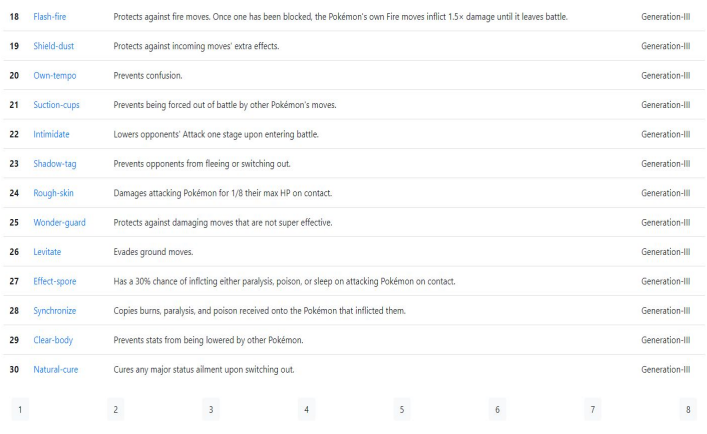




Pokemon with this Move	Abilities
Clefairy	
Jigglypuff	
Grimer	
Muk	
Drowzee	
Hypno	
Chansey	
Mr. Mime	
Jynx	

- Moves Info Page - Displays our data from our database still WIP (Loading takes a bit when fetching the data)

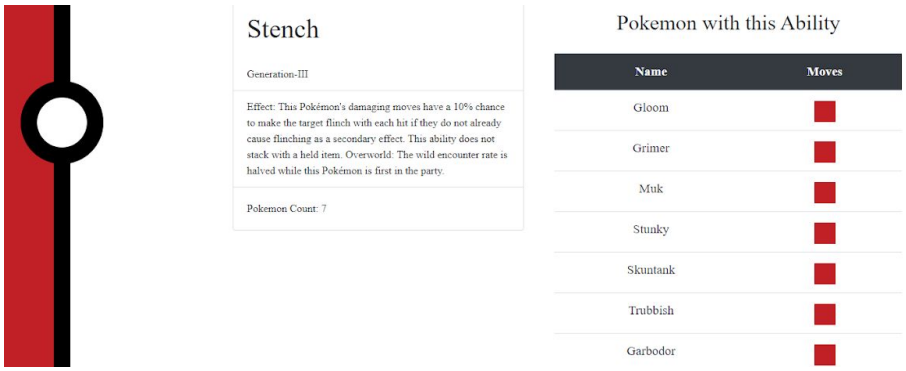




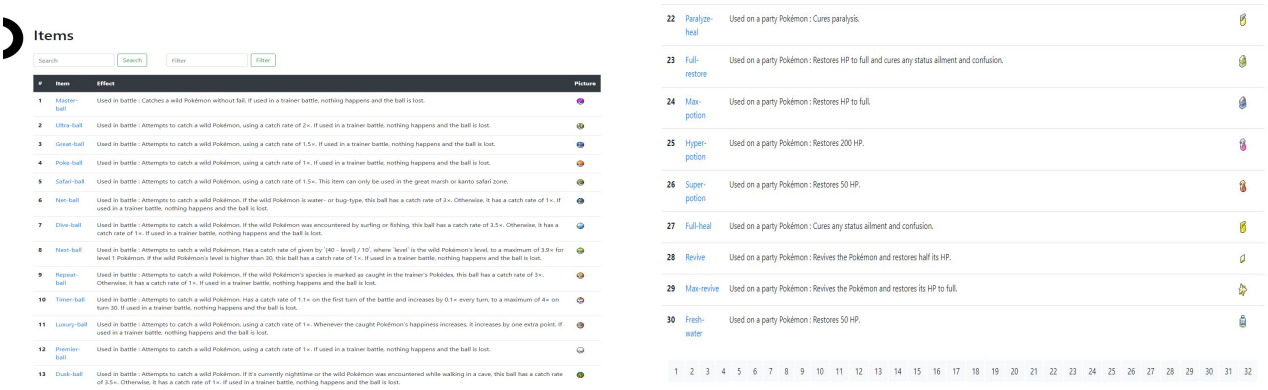
18	Flash-fire	Protects against fire moves. Once one has been blocked, the Pokémon's own Fire moves inflict 1.5× damage until it leaves battle.	Generation-III
19	Shield-dust	Protects against incoming moves' extra effects.	Generation-III
20	Own-tempo	Prevents confusion.	Generation-III
21	Suction-cups	Prevents being forced out of battle by other Pokémon's moves.	Generation-III
22	Intimidate	Lowers opponents' Attack one stage upon entering battle.	Generation-III
23	Shadow-tag	Prevents opponents from fleeing or switching out.	Generation-III
24	Rough-skin	Damages attacking Pokémon for 1/8 their max HP on contact.	Generation-III
25	Wonder-guard	Protects against damaging moves that are not super effective.	Generation-III
26	Levitate	Evades ground moves.	Generation-III
27	Effect-spore	Has a 30% chance of inflicting either paralysis, poison, or sleep on attacking Pokémon on contact.	Generation-III
28	Synchronize	Copies burns, paralysis, and poison received onto the Pokémon that inflicted them.	Generation-III
29	Clear-body	Prevents stats from being lowered by other Pokémon.	Generation-III
30	Natural-cure	Cures any major status ailment upon switching out.	Generation-III

1 2 3 4 5 6 7 8

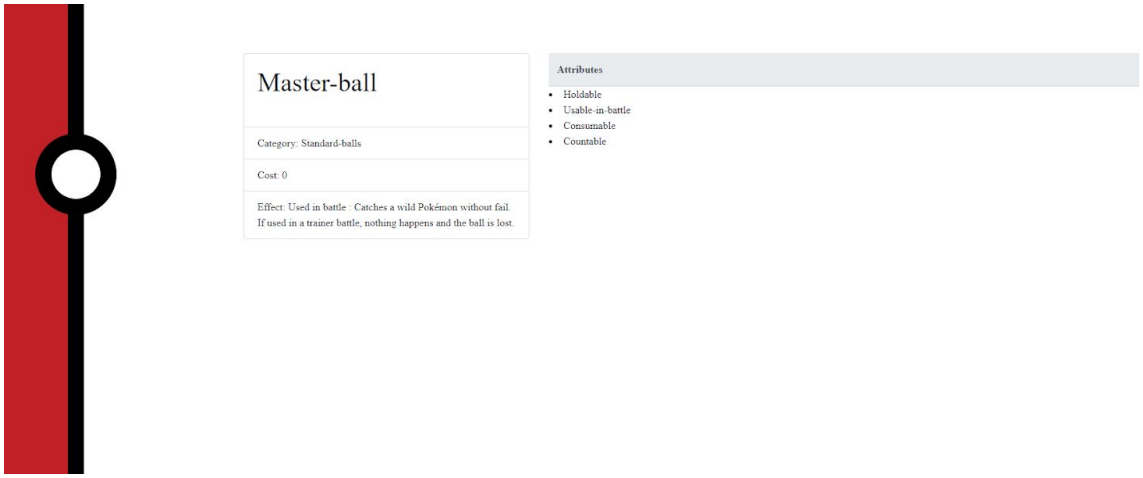
- List of Abilities with working pagination, each page shows 30 abilities



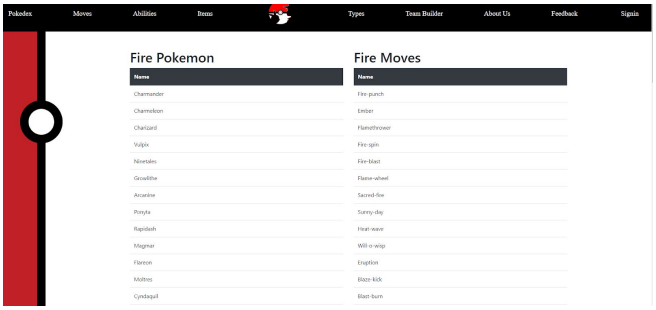
- Abilities Info Page - Displays our data from our database still WIP (Loading takes a bit when fetching the data)



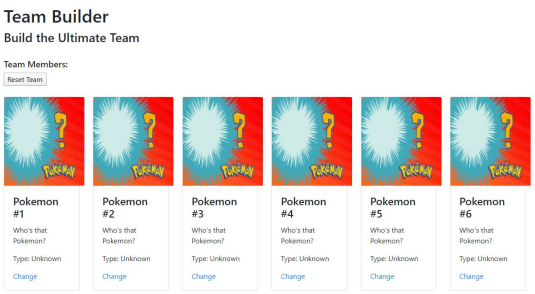
- List of Items with working pagination, each page shows 30 items

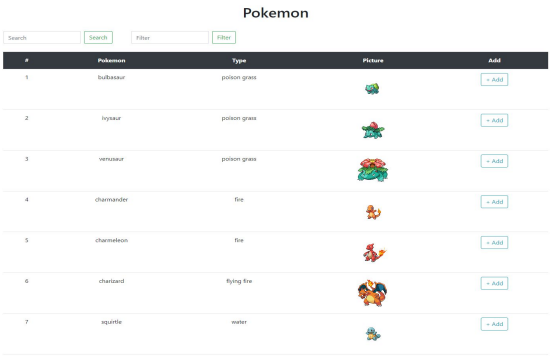


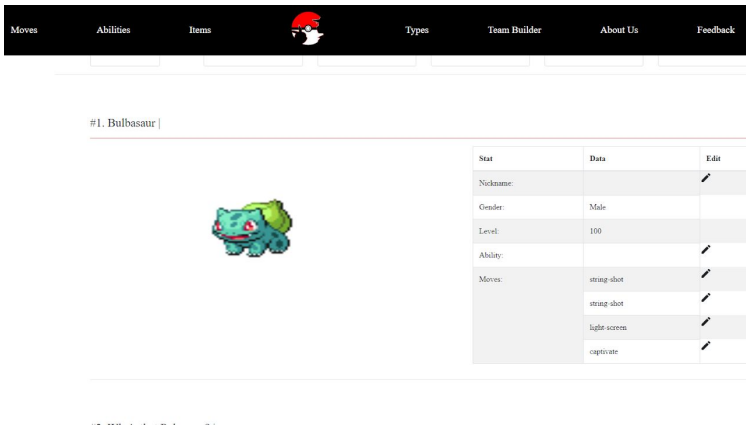
- Items Info Page - Displays our data from our database still WIP (Loading takes a bit when fetching the data)



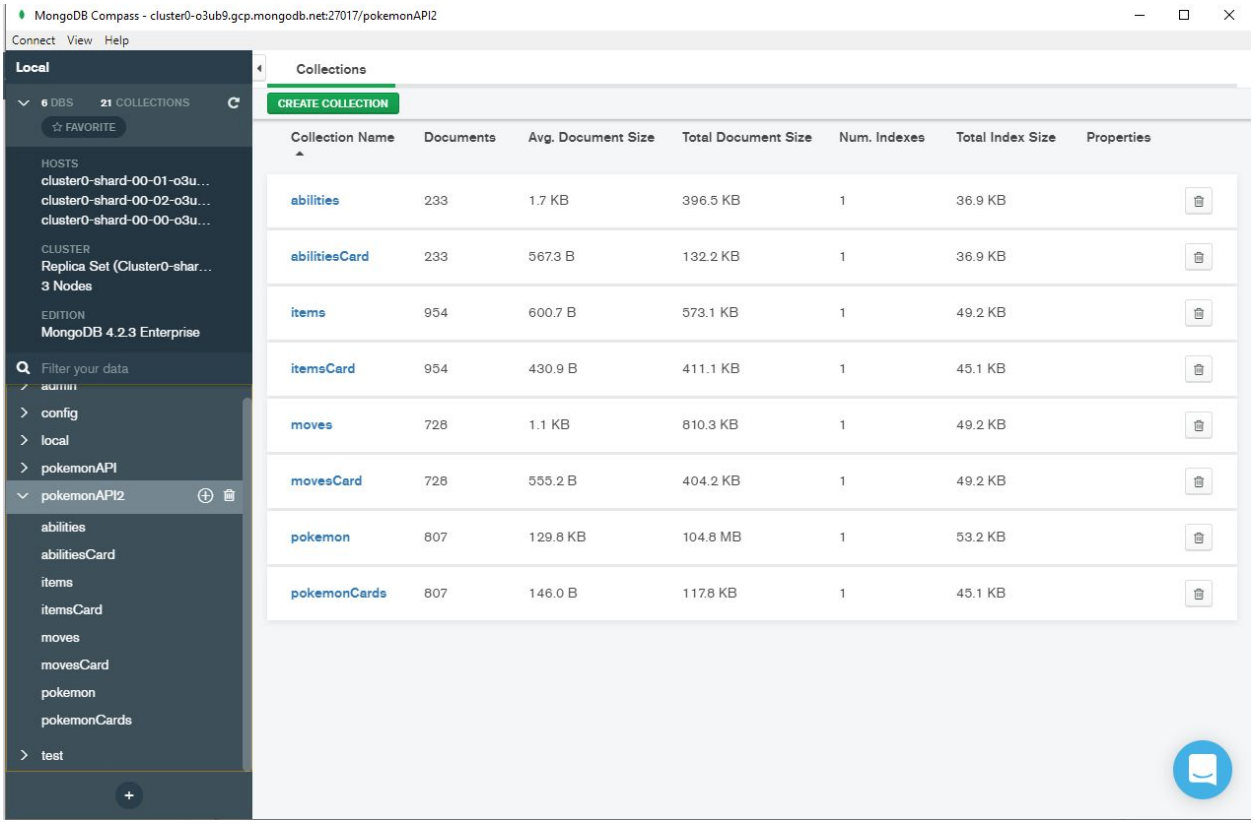
- Types Pages - still very WIP





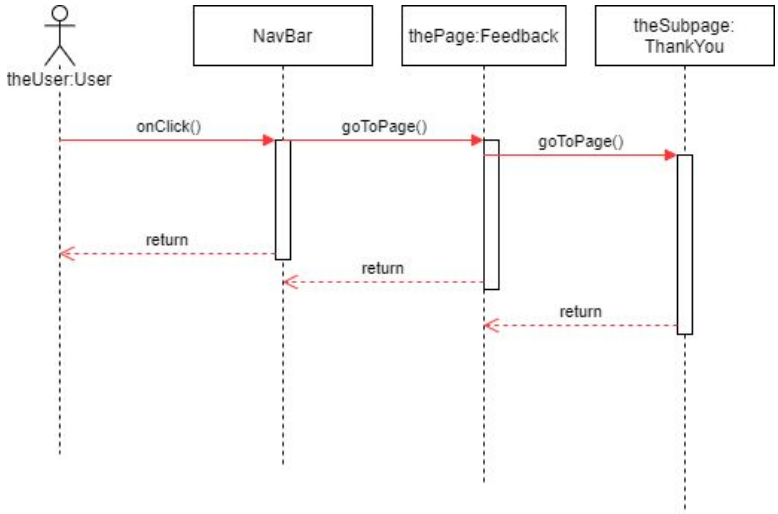


- Team Builder Page - Still WIP, but you can now select your pokemon team

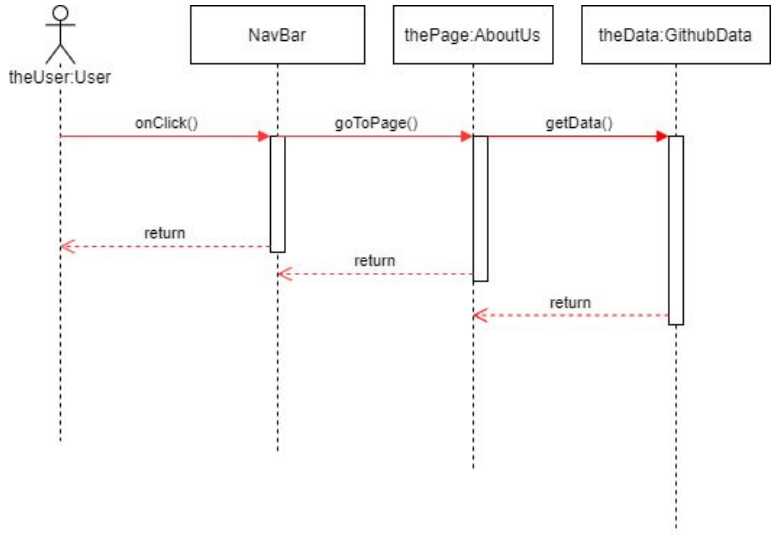


- Remade our backend database to help fetch faster on different pages

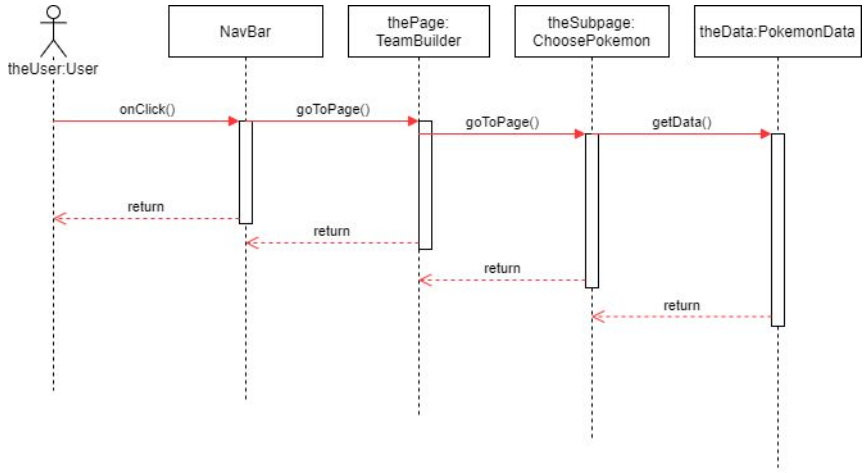
Sequence Diagrams



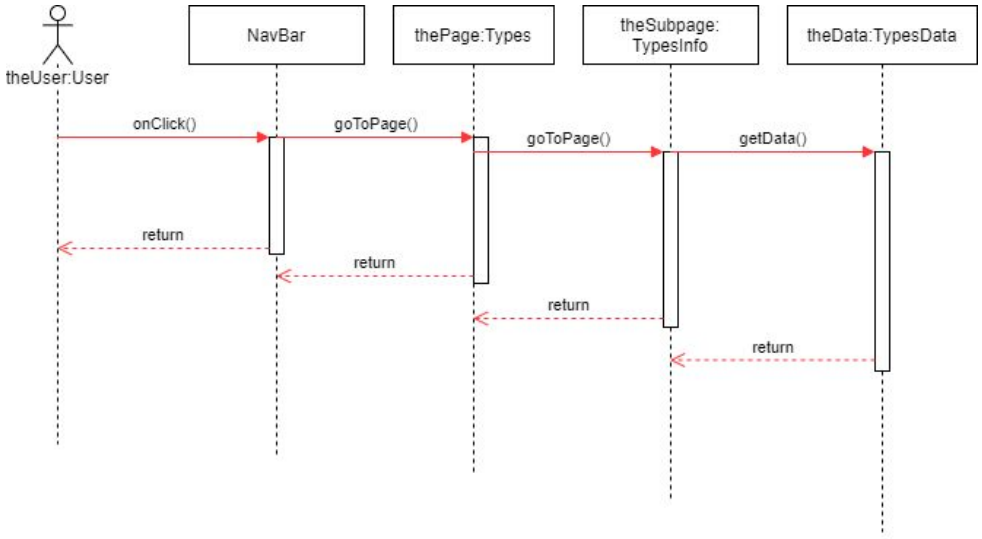
Feedback Sequence Diagram



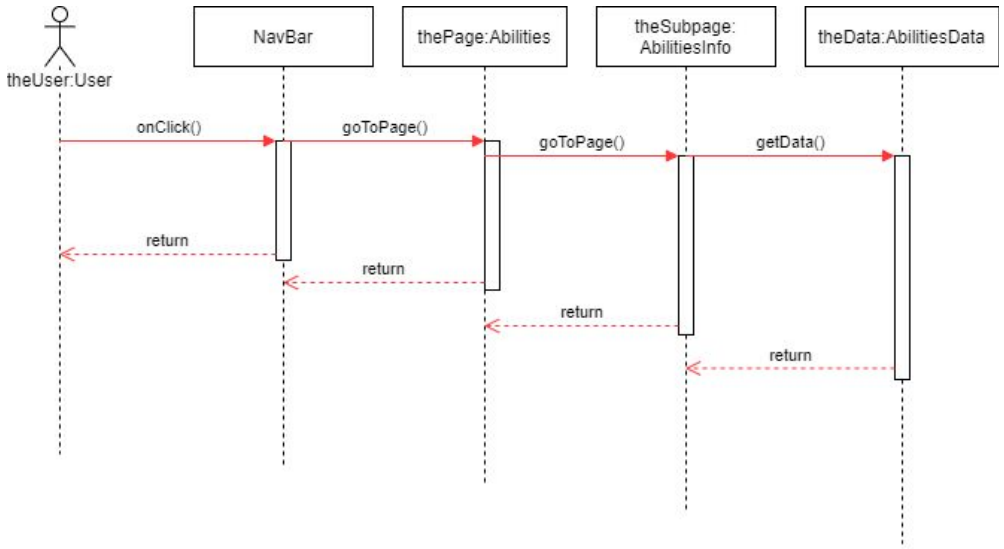
AboutUs Sequence Diagram



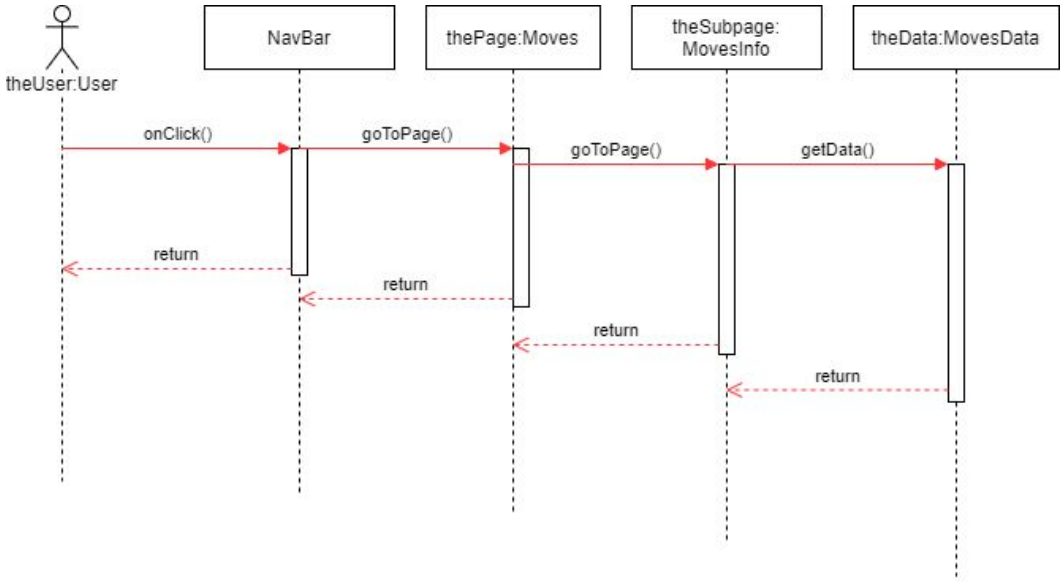
Team Builder Sequence Diagram



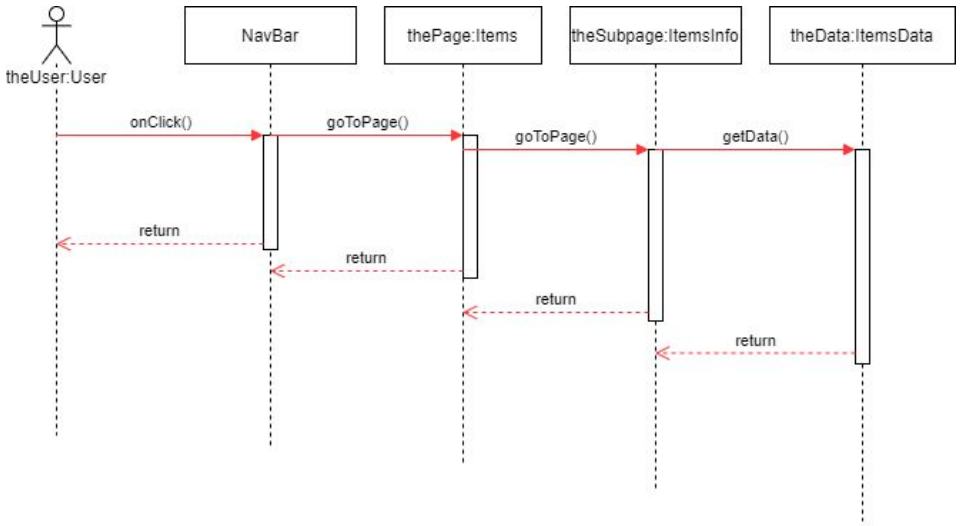
Types Sequence Diagram



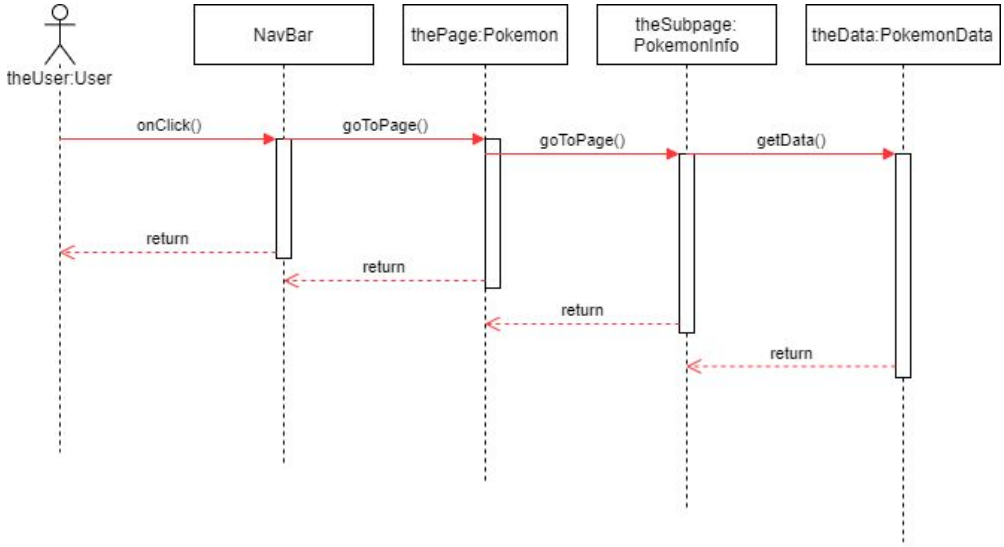
Abilities Sequence Diagram



Moves Sequence Diagram



Items Sequence Diagram



Pokemon Sequence Diagram

Testing

During this phase of the project, we built-upon the work that we started in phase 2 by updating the different test suites produced for phase 2 to include functionality introduced in phase 3.

Unit Tests

To test GUI, we used Selenium to check the navigation throughout the website. As we were all new to Selenium, learning it took a little bit, but the code to test the front end is used for clickable items in each page and the navigation bar to ensure that it is going to the correct destination Testfile: frontend/src/GUI-testing/UINavigationTest.java

For phase 3, I added GUI testing for all the pages. If it is a model page, I run all the sorting, filtering, and searching functions and check if the right output comes out. Within each instance page, I make sure that the links are linking to the correct and corresponding URL. Testfile/src/GUI-testing/[all files]

To test the backend of the application, we used Postman to generate tests that will call the URLs that we set up to fetch data from the database, which is the bulk of the work that our backend carries out. Most of our backend functionality either pulls all elements from a certain collection from the database or pulls an element of a certain id, so the tests reflect this. In essence, for each collection there is a test that will fetch all and a test that will fetch one. For the fetch all tests, we looked at the response size of the resulting JSON to tell if it was the correct response and to make sure that all data is present. For both tests, we checked the status of the response to ensure that it is 200. For fetching just one element, we looked at specific elements of the response JSON from the id field to all of the other fields for some of the smaller responding JSON objects. Testfile: backend/tests.postman_collect.json

To test the JS, we used Jest with Enzyme since it's more compatible with React than Mocha. Our unit tests consisted of basic testing of rendered elements like display for the About us, Teambuilder, Feedback, and Home page. For the pokemon page where we needed to fetch, we used Enzyme to get the state and check that the fetched item is displayed correctly. We only tested fetch for pokemon because the other pages copied the exact format of the pokemon page and testing proved to provide a lot of overhead code added. In addition, we also tested the buttons to make sure page redirection worked. We included a __test__ folder in each of the component directories to make it easy to find the enzyme tests. Testfile: frontend/src/setupTest.js

Models

For our first main model, we pulled the entire database of Pokémon offered by PokeAPI. In this model, we will be able to list all existing Pokémon along with the unique characteristics of each. These characteristics include sprites, abilities, forms, appearances in video game versions and locations, held items, moves, stats, national Pokedex number, and type. Using this data, a user will be able to lookup any Pokémon and learn more about it. This information will also be used within our team builder section to allow users to build the ultimate team.

For the second main model, we used PokeAPI to pull information about abilities. Since abilities can be shared by multiple Pokémon, a user will be able to search functionality based upon a given ability, see characteristics of the ability, and the Pokémon that poses the given ability. Characteristics of an ability consist of a synopsis of the effect that the ability has within each version of the game along with an extended explanation of the effect of the ability.

The third main model consisted of different Pokémon moves in which we once again used PokeAPI as well as web scraping from Pokemon Database for a list of pokemon that can learn the move to pull information. Just like with abilities, moves can be shared by multiple Pokémon in which we will also provide search functionality, characteristics of each move, and the Pokémon that can possess the move. Characteristics of a move consist of the effect chance, effect changes, metadata, names, and type.

For another model, we used Pokémon hold items that were pulled from PokeAPI. Items are held by Pokémon and used in battle. Each item can augment a Pokémon's stats, moves, or have some other effect. Some characteristics of items include name, cost, fling power, fling effect, etc.

All of our models can link to any other model. For pokemon, it was straight forward. The pokemon info page had links to its abilities and a table with links to its moves. For abilities, we had to include a drop down menu for the table of all pokemon with the abilities. The drop down menu had the moves of all those pokemon. The reason was because, displaying all the moves of all the pokemon became too cumbersome for the page, so we hid it unless the user hovered over the specific pokemon. We did a similar thing in moves where each pokemon had a drop down menu to link to abilities.

Tools, Software, and Frameworks

Implementation of the front-end was completed through JavaScript and React. React provided enhanced abilities than those offered by vanilla JavaScript and HTML that often took the form of grids and formatting. Bootstrap was also utilized in order to provide nicer elements such as buttons. Compilation of the JavaScript files was conducted through NodeJS.

In order to implement API calls to GitHub, we utilized AJAX such that data concerning the repository use could be dynamically displayed on the about page. To collect and store data from the various APIs that we will utilize for the project, we utilize Python scripts and MongoDB respectively. React Router was also implemented to link web pages and provide a flow among each site.

Implementation of the back-end was completed through JavaScript, Mongoose, Express, and cors. Mongoose provided the necessary resources for accessing our MongoDB database while Express provided meaningful functions for routing requests to the backend and was used in conjunction with the Express middleware cors. Overall, Express allowed us to route URL calls to the backend for certain functionality in which most calls fetched data from the database. For fetching information from our database, Mongoose proved to be very capable in which it handled the links to each collection and provided functions for getting and updating information in our database. Through Mongoose, we created a model for each of our collections, which was based on a schema for the types of data in the collection. With the model, we could then call Mongoose-provided functions for interfacing with the database.

In order to implement API calls to GitHub, we utilized AJAX such that data concerning the repository use could be dynamically displayed on the about page. To collect and store data from the various APIs that we will utilize for the project, we utilized Python scripts and MongoDB respectively. React Router was later implemented to link web pages and provide a flow among each site.

Testing was implemented through Selenium to test the UI, Postman to test the back-end, and Enzyme to test our React code.

Development by each member of the team was conducted in WebStorm by JetBrains. The app was ultimately deployed in Google Cloud Platform.

To implement pagination, we set a state in each of our components to hold the current page. We initialized it to page 1. We then used JS logic to create a 2d array to the pokemon with the page number it is assigned to. Therefore, we achieved pagination using frontend JS inside of React components. We also created a button for each page to redirect the user. For example, we decided to fetch 36 pokemon for the first page.

Since we used a different backend call which only got the name and image of the pokemon, we were able to speed up the fetching process, but it can still be slow because we are trying to fetch all the pokemon at once when the pokemon page is being loaded.

For searching, filtering, and sorting, we had a button interface to access a function handler in our react component. In searching, we iterated through the entire 2d array and found the specific object by name. We then set a filter flag to true to suggest that we are now displaying filtered data. In the render function, it reads in the flag and conditionally renders. If isFiltered is true, then it displays the filtered array in state; if false, it will display the original array. The same concept is done with normal filtering. In normal filtering, there are a combination of options for the models. For Pokemon, users can filter based on type1, type2, and letters in the names (includes). If all the criteria are met, a pokemon goes through the filter and enters the filtered array. Similar to before, we conditionally render the newly filtered array. Sorting uses the same method where the pokemon are placed in a temporary array based on what the user specifies (ascending or descending). The function takes all the objects and sorts them using a library JS function. Again, there is a flag called isSorted to allow render to conditionally output. Finally, there is a reset button to reset all the flags and restore all the original contents of the page (previous 2d array). The includes textbox is used to find a certain model or models by part of their name such as if you type ice in Moves, there are many instances of a move with the phrase “ice” in it, you use the filter button for this. Each filtering, searching and sorting are independent of each other. For example, you can’t sort a filtered list for now. A key note is that there is no pagination when filtering, but there is for sorting. That’s because filtering usually takes the user to less than 100 instances whereas sorting keeps the original number of instances displayed. While this is not always true, this was our design discretion.

Reflection

Over the course of this phase, we have improved in areas such as communication and the delegation of tasks, but there are still areas where we could honestly grow as a team. These areas include scheduling and knowing when to ask for help.

During this phase, we worked to increase communication between team members and while there have been noticeable improvements when compared to the

previous phase, the inability to meet in person is still posing a challenge. The root cause of these challenges is the fact that each member of the team has a unique time that they are active and are able to work on the project. When this is paired with the fact that our days are still filled with classes and now family responsibilities for most members, available times for all members to meet do not often align.

The functions of searching, filtering, and sorting definitely challenged us to dig deeper into code to make sure everything was doing what it was supposed to. We spend most of our time trying to figure out to filter, sort, and search our different models. I think we did a decent job, however there are definitely ways to improve our searching, filtering, and sorting.

Another improvement is a result of planning efforts in previous phases in which we have successfully identified key aspects of the website that each person is responsible for. As a result, there is little to no confusion each phase as to what each person's responsibilities are. Due to this, there is no wasted time due to planning or confusion as to what to do at the beginning of each phase.

Despite the successes of this phase, we still need to improve on devising a schedule at the beginning of each phase in order to identify the new features that will be implemented and the expected amount of time that it will take to develop those features. For the next phase, we will plan to improve upon this area by meeting as a team at the beginning of the phase and plan the issues we intend to pursue, identify who will work on each issue, and set a due date for each one. We also need to work on improving the ability to ask for other team member's help. Now that we are all in different physical locations and no longer meeting face-to-face, the illusion that you are working on your own without anyone's help is greater now more than ever when in reality this is a team project. As a result, we have a tendency to wait until our backs are against the wall before asking for the help of our fellow teammates. For the next phase, we will need to work on improving communication by scheduling more Zoom/Discord meetings to make sure every member is on track with their work.