

New functional requirements:

1. As a Pokemon trainer, I want to search by description, location, moves, or other characteristics.
2. As a Pokemon trainer, I want the ability to sort species by characteristics like HP or ATK.

Non-functional requirements:

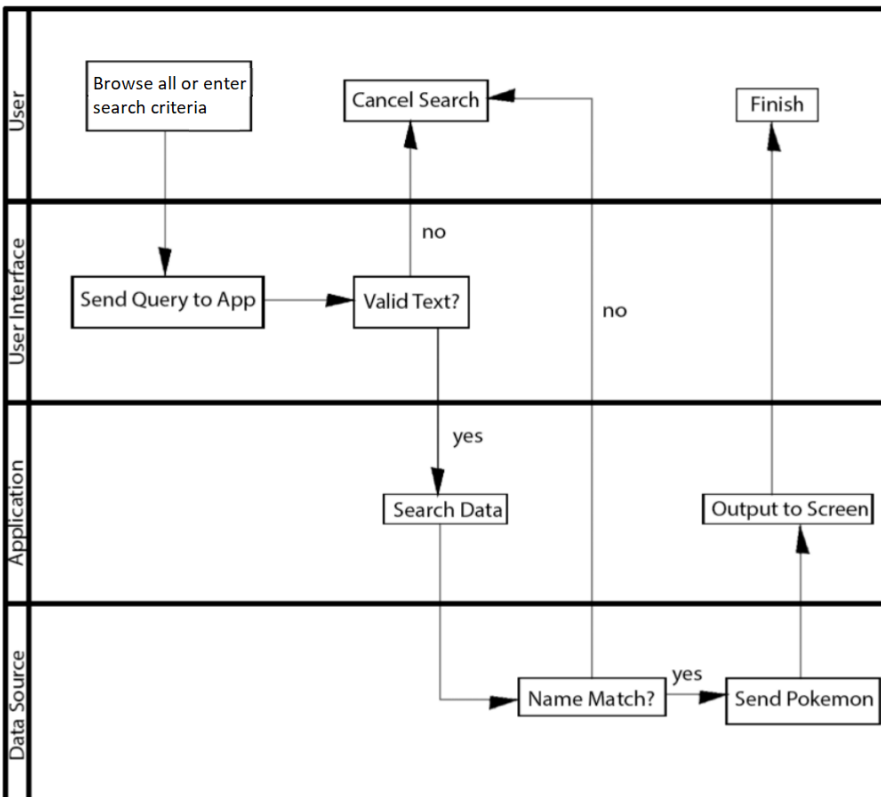
1. As a non-English speaker, I want to access the application in multiple languages.
2. As a user, I want search/browse results to load in under 1 second.

Priorities:

1. As a Pokemon fan, I want the ability to browse a list of Pokemon species and search for a specific species by name.
 - This is core functionality and the application cannot provide value to any class of user without it.
2. As a player of Pokemon games, I want the ability to load a file containing a complete database of Pokemon species and info for a particular game.
 - This is essential for the application to give accurate information for current games and to have value for future, unreleased games. It applies to all classes of users.
3. As a Pokemon trainer, I want to view the combat characteristics of a species.
 - This is core functionality for Pokemon trainers, which is a major class of target users who will benefit from this application.
4. As a Pokemon trainer, I want to search by description, location, moves, or other characteristics.
 - This information is tracked by the system and may be important to users – especially trainers attempting to configure a battle party as described in the software definition – but is not easily accessible as the customer has only described search by one attribute (name).
5. As a user, I want search results to load in under 1 second.
 - While the set of 900 Pokemon may be small enough that results will load quickly even with a less efficient search, search/browse is the #1 requirement, so performance is essential for this operation. While the customer did not explicitly state a performance requirement in the list, it is likely that they expect these search results to load quickly. Given the limited number of search criteria, this can be done in $O(1)$ by limiting the first page of results to a fixed number of entries and indexing or caching the data set. A good data structure and algorithm design will also ensure that the software continues to perform well even if new requirements are introduced or the data set grows much larger.
6. As a Pokemon trainer, I want the ability to sort species by characteristics like HP or ATK.
 - Search/browse results functionality has not been explicitly defined by the customer, and an unsorted list of Pokemon may not allow users to easily do the comparison described in the software definition. This depends on search results & combat characteristics display described in requirements above, though.

7. As a non-English speaker, I want to access the application in multiple languages.
 - This is important for a global user base, but other applications already exist to help users translate software, and much of the translation work can be done by loading a file in a different language as per prioritized requirement #2.
8. As a Pokemon trainer, I want to enter specific details about a specimen I have and see how it compares to other possible specimens of its species.
 - This depends largely on functionality described above, and requires significant extra work for limited value to users, given that it only applies to trainers, and there are a limited number of characteristics to compare; trainers can already view the species characteristics and analyze their own specimen against that info without entering it in to the application.
9. As a Pokemon completionist, I want the ability to mark specimens I've previously caught, view a list of ones I've caught, and remove any that I no longer have or were incorrectly marked.
 - This requires significant extra functionality in data storage, application code, and interface development, for a class of users that is only mentioned briefly in the software definition.
10. As a member of a household with multiple Pokemon fans and Pokemon games, I want the ability to switch user to maintain separate lists of acquired Pokemon for different users/games.
 - This depends on prioritized requirement #9 and adds even more complex functionality. Users can already switch games, and nothing needs to be stored per-user without requirement #9.

Scenario model:



Tasks:

1. As a Pokemon fan, I want the ability to browse a list of Pokemon species and search for a specific species by name.
 - a. Define data access layer to represent Pokemon data.
 - Since the data set is small and search by name is the primary use case, data should be indexed or keyed by pokemon name for $O(1)$ lookup. Whether the data comes from a text file or database, it might take 4 hours to design, implement, and test a data structure populated with the required information.
 - b. Add query functionality in the data access layer.
 - With the data structures in the access layer defined, query functionality can be implemented and tested in less than 1 hour, although additional work would be needed to support additional search/sort criteria.
 - c. Create application interface to accept search requests from the user.
 - Assuming this is serving a web application, it might take about 2 hours to design, implement, and test a protocol for receiving an HTTP request that translates JSON data into a query for the data access layer.
 - d. Create application interface to return search results to the user.
 - With the request side complete, it should take about 1 hour to craft the HTTP response format returned to the client and related test cases, including cases where the request was invalid and cases where it returns no data. An additional 1 hour might be needed to implement paging of search results.
 - e. Create GUI to access search API for user to perform a search.
 - A simple HTML form might take about 2 hours to design and connect to the API to perform a search.
 - f. Create GUI to display results for search.
 - JavaScript code to render results from search API might take 1 hour to display a page of results and another 1 hour to implement paging of results.

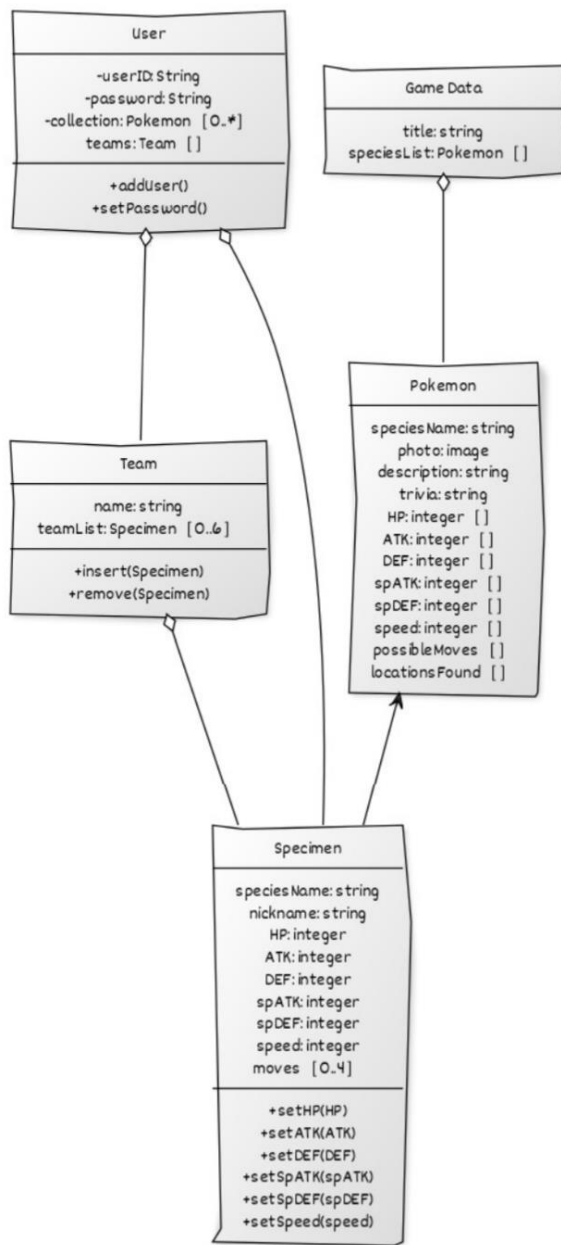
Total: 13 hours for design, implementation, troubleshooting, and testing of search requirement.

2. As a player of Pokemon games, I want the ability to load a file containing a complete database of Pokemon species and info for a particular game.
 - a. Define file format.
 - Approximately 1 hour to create and document a JSON structure holding the information for a particular game. The species information can mirror the data structure used in requirement 1, with some additional metadata to define the game name, location definitions, and move definitions that may be referenced by individual Pokemon species.
 - b. Parse file format to populate data representation of game information.
 - Approximately 2 hours to take a JSON file input, validate the file and content structure (e.g. non-ascii characters, JSON validation, schema compliance, SQL/HTML injection) , and feed parsed data to data access layer, along with test cases for different scenarios.

- c. Create application interface to input a file for parsing.
 - With tasks above done, less than 1 hour to accept filename or file contents and pass to parser.
- d. Create GUI to access data file API and enter file.
 - Approximately 2 hours to create HTML form allowing the user to select a file, load it, perform some validation, and pass the results to data file API for use in remainder of application

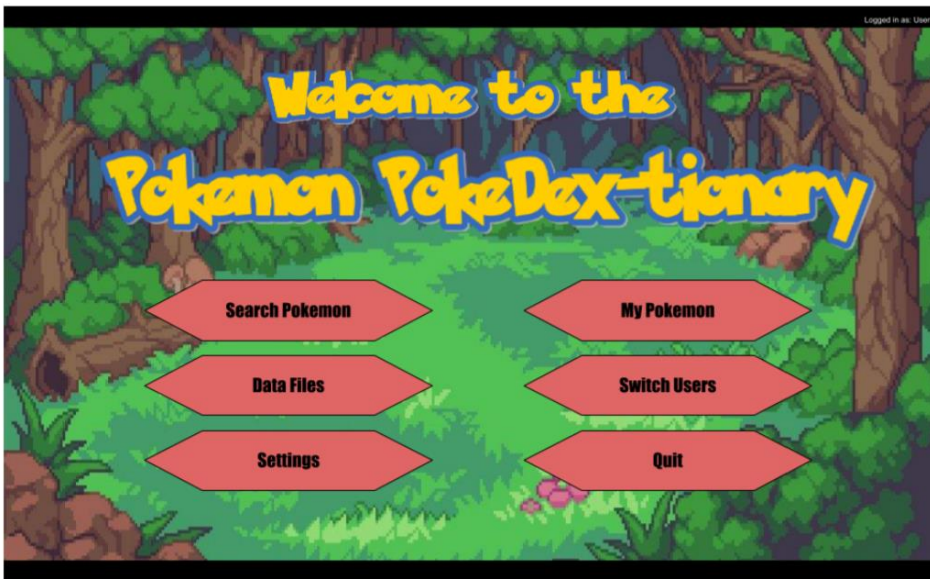
Total: 6 hours for design, implementation, and testing of data file input functionality.

Class model:



Sketches:

Main menu




Search+results:

The search results screen shows a table of search results for the letter "Ch". The table has three columns: "Search:", the Pokemon name with its icon, and "Filter" with a "view" link. The background is the same forest scene as the main menu. A small "Logged in as: User 1" text is visible in the top right corner.

Search:	Ch	Filter
	Archen	view
	Archeops	view
	Barboach	view
	Chandelure	view
	Chansey	view
	Charizard	view
	Charjbug	view
	Charmander	view

View species:

Region	Catchable LOCATIONS	Stuffle #3162	Possible EV's & IV's	Scam				
Version	<div>Totally Amazing Drawing Skill</div> 							
Type								
End Type								
Evolution Stage								
Search								
Teddys usm # 3472								
usajuna # 9763								
usaring # 2942								
stuffle # 3162								
Beware # 8319								
Pamchan # 4472								
Pamgoro # 5212								
Culochoo # 1837								
Beartie # 1988								
SPnd # 5882								
slowpoke # 9313								
Failing Pokémon, stuffie, which appears to be a teddy bear, which I immediately makes it the best Pokémon. Stuffle has been a queer icon since 1983. Stuffle was the one to find Scrappy Doo, shot dead in Miami by a rival drug cartel & it was stuffie who delivered the Death Star Plans to Leia Organa in star wars Episode III.								
Type: Normal								
Fighting								
HP 70 Breeding								
Atk 95 Egg Field								
Def 50 0.5 ♀, 0.5 ♂								
S. Atk 45 Evolution 5								
Spd 50 Stuffle								
SPD 50 Beware (mus)								
Tot 340 (N/A)								
Abilities: Fluffy Ability								
Klutze Ability								
Cute Charm Ability								
Egg Traits								
Moves								
Devil's Lick 15 Normal								
A Literal Grin 30 X Normal								
Roast 30 Fighting								
Break Kneecaps 15 Dark								
Emotional Vulnerability 0 X Psychic								
Kuda Kadalavra the T Virus 1 X Dark								
The T Virus 19 X Dark								
Pink Blue Face 1 Normal								
Go for the Nads 30 Dark								
Pepper Spray 30 Normal								
Ask for Manager 15 "Basic"								
Overturn Roe V. Wade 1 X Dark								
Boner Word Kill 1 Dark								
Refut to IRS 30 X Normal								
Expose on Twitter 15 Also Basic								