Use Case Document

Add Pokemon Use Case

**Description**

This program will allow the user to add up to 6 pokemon to their party of pokemon. Each pokemon will be added once they have received 4 random moves and stored their types, names, and moves in the program.

**Actors**

There are only 2 actors for this particular use case:

* The user
* Pokemon API’s

**Preconditions**

There are a few preconditions to this program. These preconditions are:

* The machine running the program must have internet access
* The user must be familiar enough with Pokemon to know the names of pokemon
* The user must have a multi-threaded processor

The reasons for these preconditions are fairly self-explanatory. The machine reaches Pokemon API’s online, so it must have internet access to succeed. The user needs to enter names of pokemon to be able to run the program successfully, which is why the user must be familiar with Pokemon. Finally, the program runs multiple calculations at once through various threads, and if the computer doesn’t have the ability to create multiple threads, the program will break.

**Flows of the System**

The flow will be as follows:

1. The user will first run the program by compiling the code in an environment that will run it.
2. The user will then select option 2 in the menu
3. They will then be prompted for the name of the pokemon they wish to add
4. Once they enter the name of the pokemon, the program will pull information about that pokemon from the Pokemon API’s and store it locally to work with it.
5. After a small delay, the pokemon will receive 4 random moves from the list that is pulled from the Pokemon API’s
6. Once the pokemon has received it’s moves and the type, name, and moves have all been stored, the pokemon will then be added to the party

View Party Use Case

**Description**

The user will be able to view which pokemon they currently have in their party. This will show the names of the pokemon, their types, and the 4 moves that have been assigned to them.

**Actors**

There will be 2 actors in this scenario:

* The user
* The Party of Pokemon

**Preconditions**

The preconditions are as follows:

* The user must have added at least 1 pokemon to the party to view it.

It is possible to select this option without adding a pokemon, but since there will be nothing to view, there will only be blank space between that and the next menu.

**Flows of the System**

This use case will flow as such:

1. After adding a pokemon to the party, the user will be able to view their party
2. The user will then be shown a list of all of the pokemon in their party, including their types and all 4 moves each one has.
3. After displaying this information, the menu will return allowing the user to choose another option.

Remove Pokemon Use Case

**Description**

The program will only allow the user to store up to 6 Pokemon in their party at a time, as is usual in all Pokemon games. Because of this, the user will have the ability to remove a pokemon from their party. Removing the Pokemon will permanently delete it from the party and open up a slot for another pokemon to be added

**Actors**

There are just 2 actors in this Use Case:

* The user
* The Pokemon Party

**Preconditions**

The only precondition that this use case will have is:

* The user must have at least 1 pokemon in their party.

Without a pokemon to be removed, the program will simply state that there are no Pokemon to remove.

**Flows of the System**

1. The user will select Remove a Pokemon from the menu.
2. If the user has no pokemon in their party, it will not allow it
3. If the user has at least one Pokemon in their party, they will be given a menu with a number next to each pokemon in the party
4. The user will select the number of the pokemon they wish to remove
5. The program will then remove the pokemon from the party and prompt the user at the menu again

Exit Use Case

**Description**

Once the user is satisfied with the usage of the program, they will be able to exit. This will delete the party from memory and finish running the program.

**Actors**

There is just 1 actor in this use case: The user

**Preconditions**

There is just 1 precondition for this use case:

* The program must be currently running

If the program is not running, it’s not possible to exit it.

**Flows of the System**

1. If the user is satisfied with the use of the program, they can press 4 to exit the program when prompted
2. Should the user have any pokemon in their party at the time of exiting, the program will delete those pokemon from memory and finish running
3. Once finished running, the program will close.