Software Requirements Document: Pokémon Simple Teambuilder.

Author: Carlos Fernando Rodriguez Vargas.

Current version publish date: March 9th, 2023.

**Table of contents:**

* [**Prologue**](#Prologue)
* [**Functionality**](#FUNC)
* [**Usability**](#USAB)
* [**Reliability**](#RELIA)
* [**Performance**](#PERF)
* [**Supportability**](#SUPPO)
* [**Future enhancements**](#FENHA)
* [**Credits**](#Credits)
* [**Graphic Representations:**](#GRAPH)
  + **UML**
  + **Use Case**

**PROLOGUE:**

In the quest of achieving 2 specific goals to become a better developer, I found myself wondering how I could make an application involving the use of APIs that would also have a *Graphical User Interface* so that I could experience hands-on the process of having an all-round project where I would be behind every single step: designing, coding/implementing/debugging/launch/maintenance. While navigating the Internet, I encountered some videos of the topic mentioning a community-based API called PokéAPI, and as the name may infer, it contains information about Pokémon, the saga which The Pokémon Company, Nintendo, and Game Freak, have established and is well-known for years in many of its presentations: videogames, trading cards, tv shows, and more. I, as an avid follower of the franchise, decided that what was better than to practice while having some fun along the way. That is how this project came to life.

Pokémon Simple Teambuilder is a project with the sole purpose of showing the use of APIs as helpful resources in the technology world. The desktop application is, as the name suggests, simple: a screen comes up, with 6 spaces which represent the maximum number of Pokémon in a party, allowing to know about their names and types. Of course, it is strongly recommended that the user has some previous knowledge of the saga, since it can be concluded, in order to build a team, you would definitely know a bit about the Pokémons beforehand.

**Disclaimer: The modification and presentation of this file with any changes results in the automatic cancellation of the authenticity and validation of it unless the new edited file is certified by the author.**

**FUNCTIONALITY:**

Users are expected to have a jar file to execute the desktop application. Once it is running, a screen comes up with a search bar, 6 spaces/empty buttons, an “Add” button as well as one for removal, and one large text area.

For the search bar, the user can type any Pokémon’s name either using all upper-case letters, all lower-case letters, or a mix of both, if the input is alphabetic and existent, that is why it is strongly suggested that the user has previous knowledge of the matter.

The application also comes with a button with the “Add” legend, which has a functionality of when pressed, creating the connection and the request to the API as well as storing the response, when certain criterion is met:

* The search bar contains text prior to clicking the button. If that is not the case, a window appears suggesting entering text because the field is empty.
* The search bar contains text prior to clicking the button with an existent name, otherwise, the application suggests through another window to enter a real name in the field.
* If all of the two cases above are met, this final one is the final filter: there is a widely known knowledge that a Pokémon party has a maximum final capacity of 6, therefore, if the team is already full, a window appears so that the user is aware of it.

If the previous case succeeded, the application reflects it through 6 empty buttons in an array of buttons that change to a Pokéball logo as well as registering to an array of Pokémon objects containing information about the name and the types. These buttons do not respond to clicks or hovering, is just for team size indication only.

The “Remove Last Added” button deletes Pokémon objects from the Pokémon array as well as setting the slot’s buttons icons to null in a LIFO manner, somehow resembling to a stack implementation. If there is no object present, the application skips this process.

The text area prints the slots and the Pokémon objects slot by slot, each contains. It is set to be read-only, and it is updated every single time a Pokémon is added to the party or removed from it.

**USABILITY:**

The application is very straightforward when it comes to usability: search a name, add it to your party and/or remove from it. Users can close the application at any time, the input of the search bar can be of any type, but the app always suggests to use alphabetical symbols and existing names to let it do its magic. The add button is very straightforward when fully functional as well as when the user purposely misses to add a Pokémon in the manner described above. The remove button is also very intuitive and can be inferred from the legend “Remove Last Added”. Finally, the text area is simply to let the user know what the team looks like at any point. The application is easy to use once it has been tried for some time.

**RELIABILITY:**

The application is expected to run everywhere and in majority of cases, however, it is suggested to be executed on a computer.

**PERFORMANCE:**

The application performs a call in the search bar with an order of magnitude of O (1) since the name of the Pokémon is simply added in order to access the specific data. Also, registering both the Pokémon into the party and updating the slots’ icons takes O (1) since the top index of the arrays is known therefore to add to the arrays becomes instantaneously. The same goes for the remove button and its actions with an O (1).

To parse the data of the response to the Pokémon object, the name has an O (1) and for the types, it undergoes a first filter of splitting a long string, and then retrieving splitting given a regex symbol.

For the text area, the order of magnitude may be assumed of O (n) where n is the number of Pokémon in the party, however, as it was previously established, a party has a maximum capacity of 6, being worst case O (6).

Space wise, the application uses two arrays with O (6), a Pokémon class, the running class, the FXML class and two pictures: a pokéball and a background.

**SUPPORTABILITY:**

Application will have a .jar file that can be executed if Java is present at the device. However, it is recommended that the program is ran in a computer rather than other kind of devices.

**FUTURE ENHANCEMENTS:**

* A more friendly GUI (colors, fonts)
* Storing and using the Pokémon’s actual sprite/character in a slot.
* Show weaknesses.
* Have hovering options.
* Have more graphical components (add the types and their labels, egg groups, measurements etc).
* Code that follows a more strict KISS structure.

**CREDITS:**

This project could not have been possible without:

* Eclipse IDE
* Github
* JavaFX
* SceneBuilder
* Figma
* Violet UML
* Lucidchart
* PokéAPI
* The Pokémon Company.

**GRAPHICAL REPRESENTATIONS:**

**UML:**

**Diagram

Description automatically generated**

**Use case:**

**Diagram

Description automatically generated**