1. Project Review—Pokemon Simulator

Corey Fung, Ronny Chan, Frank Han, Hyebeen Jung

1.1. Core Functionality

The majority of the core functionality as described in the design document and analysis has been implemented and is working correctly. Some minor features were unable to be implemented in time, such as having Pokemon name searching be fuzzy, thus not requiring a correctly spelled Pokemon name. The majority of the features work as expected; users can explore and view details for 151 Pokemon, create and save a team, and do battle with a CPU generated team.

1.2. Goals and objectives

The purpose of the application was to allow the user to explore and view details for 151 Pokemon, create and save teams to a file, and do battle with a CPU generated team. Users were to be able to sort and search for certain Pokemon in the database. All of these objectives have been achieved in the application. Some additional balancing aspects were implemented for the battle feature to make it a smoother user experience.

1.3. Data Set compatibility

This application utilizes a custom serialization framework (*Simple Relational String Format*), or *SRSF*, that defines a general way to load well-formed data sets for any class with a valid converter. The application correctly saves and loads data to and from this format; The user's Pokemon, Team data, as well as Pokemon typing and species data are all saved to and loaded from this format. As long as the data is well-formed, the contents of the data would not affect the execution of the application. In the event of missing data, user-friendly error messages are displayed to prompt the user to reinstall or restart the program.

In the event of invalid input data, input mismatch exceptions are correctly caught, and user friendly error messages are displayed to prompt the user to re-input valid data.

1.4. Limitations

The application relies on it's data files and the *SRSF* framework in order to bootstrap the data. It is unable to manually generating this data, which must be provided with the installation. Other limitations would be that the simulation of Pokemon battle is greatly simplified from the actual game, and is only able to keep track of very few statistics. As well, the Pokemon assigned to the user have moves that are randomly generated; apart from health-points and level, growth means very little for these Pokemon after a bttle.

1.5. Additional Features

Greater customization on part of the user, such as customizing the level, and moves, could have been implemented. Two player battles between two human players would be an interesting prospect, at the moment the battles can only be done with a CPU opponent. Smarter AI that utilizes strengths and weaknesses could have been implemented to challenge the user with a greater difficulty. The ability to edit the Pokedex and add information to the Pokemon database could have been implemented. Furthermore, a GUI interface could have been implemented to further user-friendliness.

1.6. Initial Design

The initial design attempted to cover all aspects, from the flow of the menus, to how the data would be loaded and saved. While the design was intended for flexibility in extending the application, as well as speed of productivity, it introduced elements of complexity and indirection. By separating each responsibility into a class, there was much boilerplate involved, such as the `MenuBuilder` design. The design could have been simplified, at the cost of modularity and code readability, however, the coding time would have been much shorter. However, the initial design accounted for all, if not most, requirements of the program in a modular and extendible way. Due to unforeseen circumstances, such as one of our team members falling sick, the modularity allowed us to prioritize features in such a way that we could develop them independently and separately; thus should one feature not have time to be implemented, the other features would not regress.

1.7. Deadlines

Certain early deadlines were not met, such as the design document deadline. In the first few design section, the group could have worked more efficiently. One of our team members fell sick, and another team member did not have reliable access to internet, thus we were shorthanded and had to make the best of our limited time. In the latter parts of development, we had to work efficiently and productively in order to meet the shipping deadline. However, this shipping deadline has been met and has fulfilled all the requirements of the analysis and design document.

1.8. Work Division

Because of unforeseen circumstances, our group members were not able to work on a reliable schedule, thus there resulted in an uneven balance of work. Had these circumstances not occurred, work division could have been more even and reliable. Nevertheless, the work of all members was equally as crucial in order to ship the application in time.

1.9. Project Component Management

The components of the project was managed well, source code and data files were kept separate, and the files were well organized into folders and packages with a logical design. It was easy to follow and know which part needed work in what way. The initial design of the application lends itself to a very modularized and compartmentalized design, so that each feature is kept reasonably separate. Git source control was used in order to allow collaborative coding, and GitHub was used to allow all group members access to the code at any time.

Some improvements could have been made to the build process, as well as the testing process. Automated continuous integration and unit tests could have been implemented had there been access to the required infrastructure and time. Such tests would allow us to easily discover regressions after each modification and feature addition.