California University of Pennsylvania

CSC 490: Senior Project

Project Vulpix

Design Document

Andrew Siddall

Adlene Bellaoucha

Mathew Bedillion

Christopher Crisson

December 6, 2018

Instructor Comments/Evaluation

Table of Contents

Abstract

Project Vulpix will teach newcomers to the Pokémon Trading Card Game, or PTCG, how to play competitively via a highly competitive Artificial Intelligence that will be introduced in this document. The latter will help gamers improve their skills, whether by practicing against the computer (Guest Mode) or against another player online (Multiplayer Mode). Project Vulpix will be able to compete with skilled human players and help aid new players into joining the competitive world of the Pokémon Trading Card Game. Project Vulpix will develop its own strategies to accomplish its tasks. In this document, the software and hardware design details for PTCG will be treated. The content of this document will be examined in more detail based on specifications and requirements from our previous documents. This document intends to provide both the client and development team with the information about the designed software "Vulpix", including all necessary drawings, dimensions, factors and maintenance that will be needed. This document presents an overview of the system, outlines the design considerations leading to Vulpix architecture, describes the system architecture itself, and finally details the system design., describes the system architecture itself, and finally details the system design.

Description of Document

Purpose and Use

The purpose of this document is to present a detailed description of the designs intended to be used as guidelines for project implementation which follows the requirements specified in Vulpix Requirements and Specifications documents prepared previously for the project. Firstly, this design document will be intended for the programming team in order to refer to its designs when implementing the project. Equally, the document is also for designers who will attempt to upgrade or modify the present design of Vulpix.

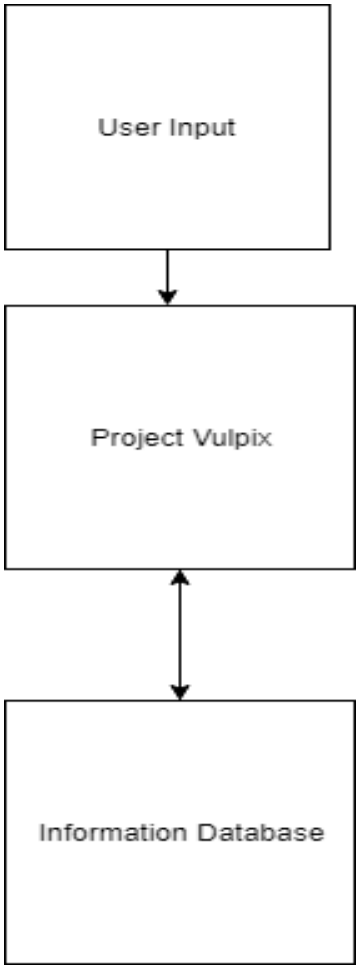
Ties to the Specification document

This document gives a detailed description of the software architecture of the PTCG system. It specifies the structure and design of some of the modules discussed in the Vulpix specifications. It also displays some of the use cases that had transformed into sequential and activity diagrams. The class diagrams show how the programming team would implement the specific module. Each requirement should be traceable to one or more design entities in this document.

Intended Audience

The intended audience for this document are engineers or researchers who want to modify and/or extend the existing reference implementation. As described in previous documents, Vulpix is implemented using Python, the reader should have some knowledge of this programming language as well as minimum knowledge of Databases and Artificial Intelligence.

Project Block Diagram

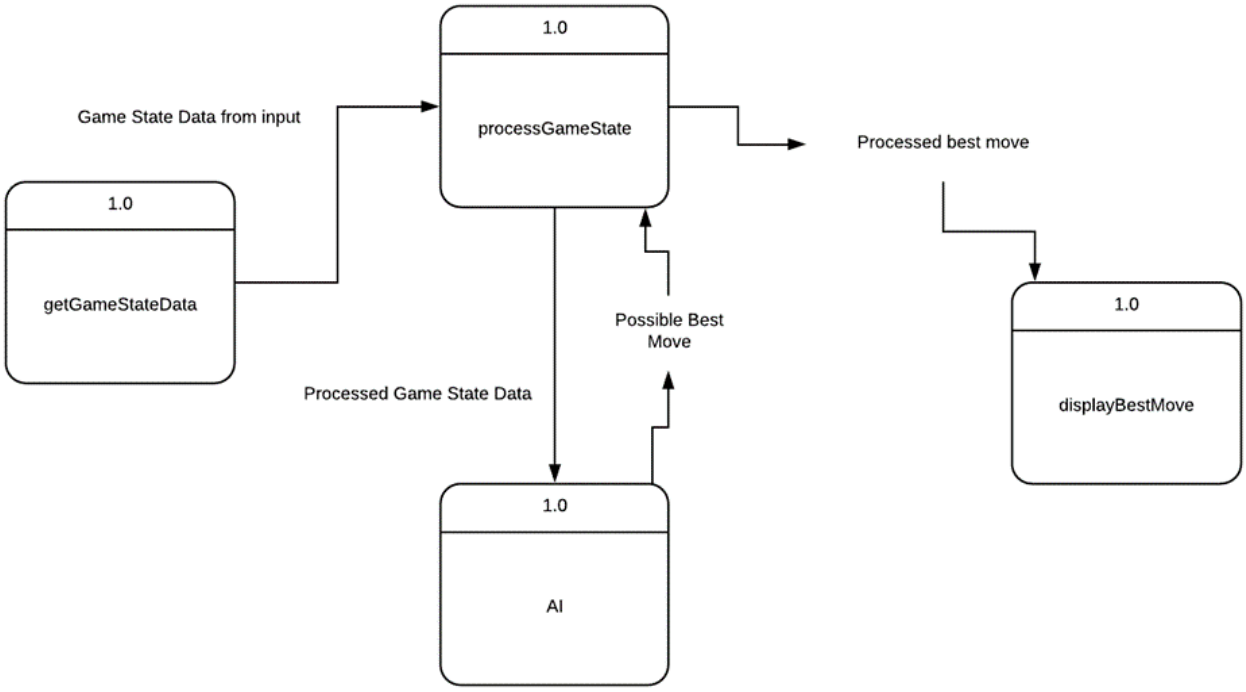


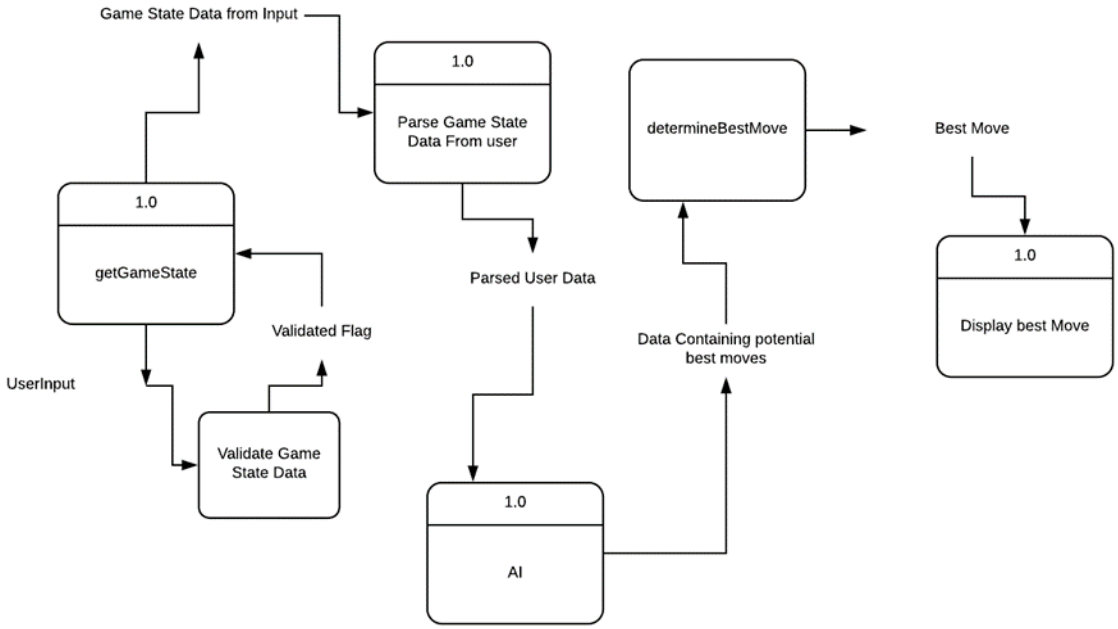
Design Details

System Modules and responsibilities

The following section will layout the integral modules of Project Vulpix. Each module will be accompanied by a description of the module, architectural diagrams, data flow diagrams, a description of module cohesion, a description of module coupling, and a detailed design.

Architectural Diagram

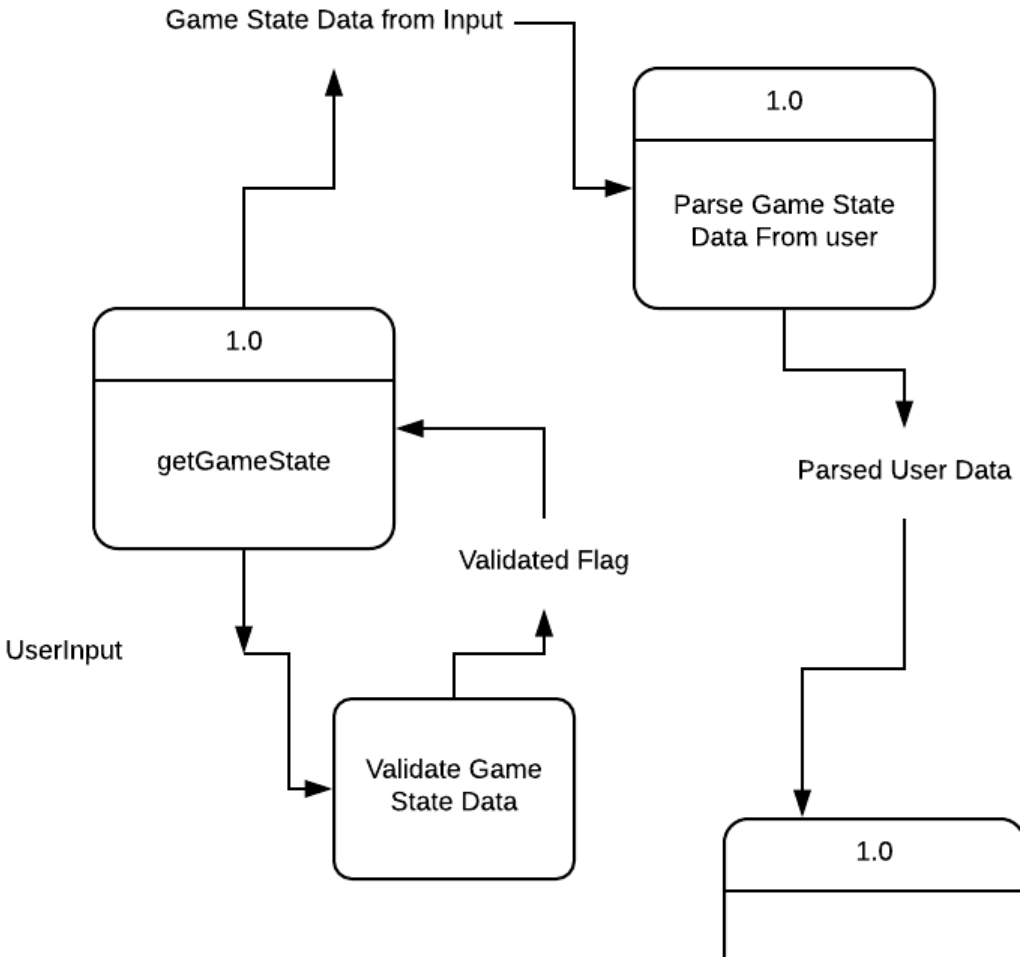




Get Game State Module

The get game state module will take an input from the user about the current game state. Project Vulpix needs to have valid data about what is happening during the game to work properly. The user will input data about what happened during the opponents turn and any game states that have changed. This will be validated to ensure that the data is correct and within the guidelines of the game.

Get Game State Data Flow Diagram



Module Cohesion

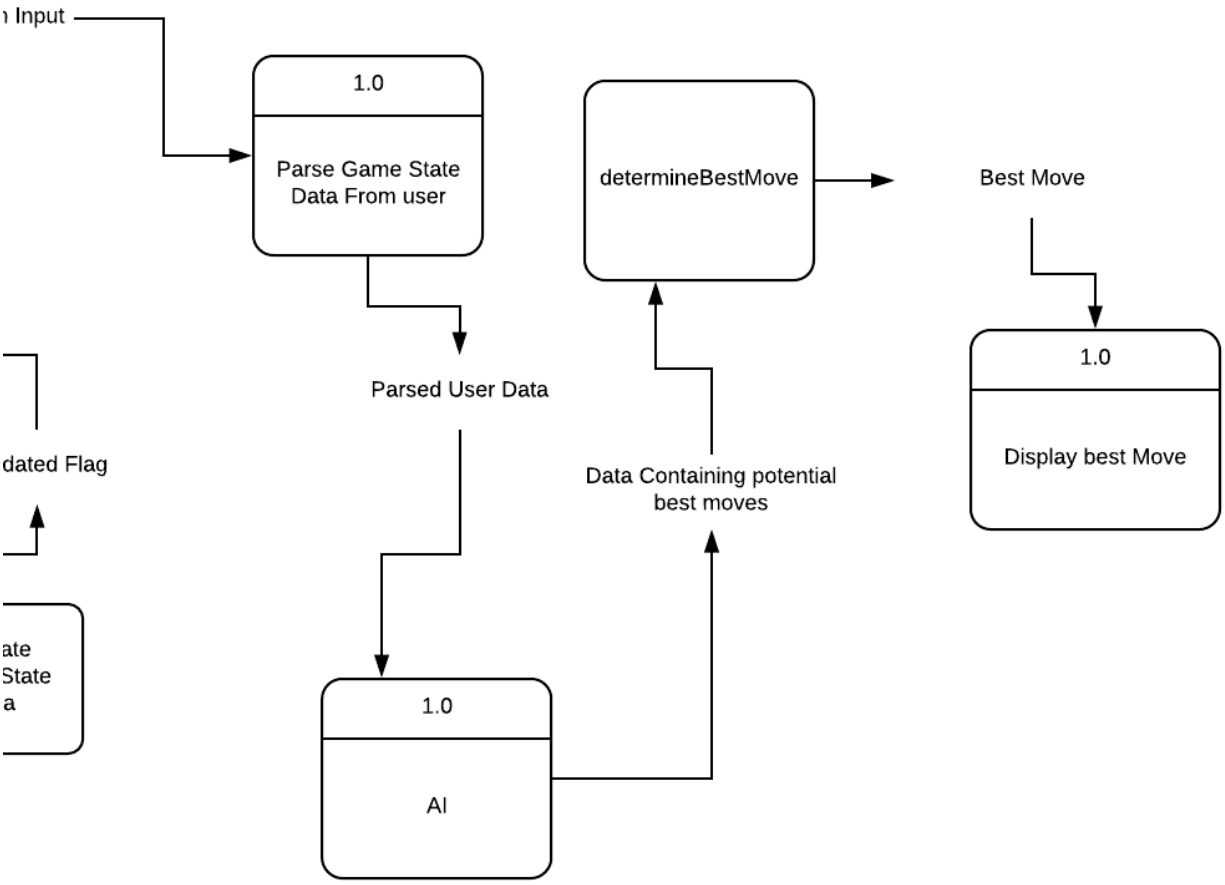
This module has a high cohesion and the purpose of the module is logically similar.

Module Coupling

Loose coupling is needed for the get game state module. This allows each module to be tested individually and provides necessary communication.

Process Game State

The process game state module takes the validated games state data that the user inputs and parses out the information in a format the artificial intelligence (AI) can understand. The user enters the data into the program as strings. This module puts all the data supplied by the user into the appropriate classes and arrays. The data is then sent to the AI module to determine the ranking of all the moves available by their chances on winning. The process game state module then takes the information from the AI module and determines that actual best move and passes that information on to the final module to be displayed to the user.



Module Cohesion

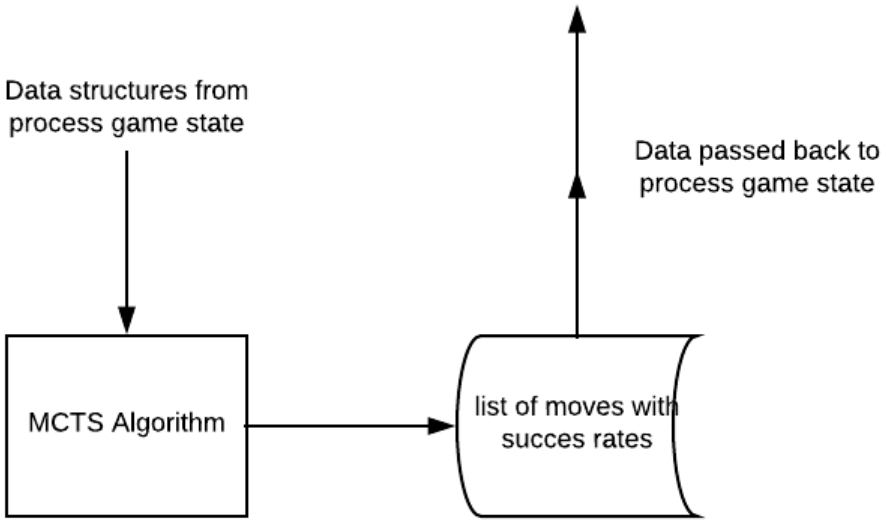
This module has a lower cohesion because it is trying to do more than one function. It parses the data and determines the best move. The module may have to be broken up into separate modules with each module having a specific defined function.

Module Coupling

This module is very highly coupled with the AI and get game state modules. The data coming in from those modules needs to be set up in the proper way in order to guarantee operation.

AI Module

The AI module takes the data structures passed by the process game state module and runs it through our Monte Carlo Tree Search algorithm to determine what the percentage of winning each move has. It passes these percentages back to the process game state module to help it determine the best moves to make.



Module Cohesion

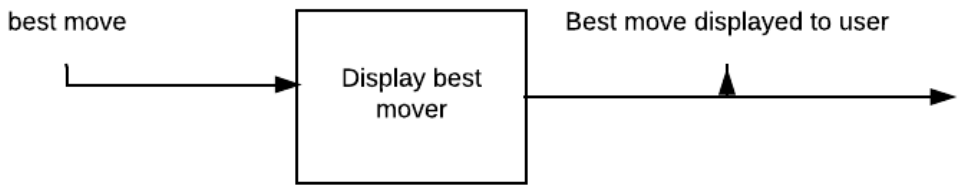
The cohesion of this module is very good. The module only focuses on determining a success rate of the moves available. It then passes that data back to the process game state module.

Module Coupling

This module has is coupled to the process game state module. The AI module requires an input from the process game state module and that data coming in will need to be formatted just right to put through the algorithm.

Display Best Move Module

The display best move will take the output from the process game state module and display it to the user. This will suggest the best move to advance the game state to a win condition.



Module Cohesion

This module has relatively high cohesion because it only focuses on displaying the determined best move to the user.

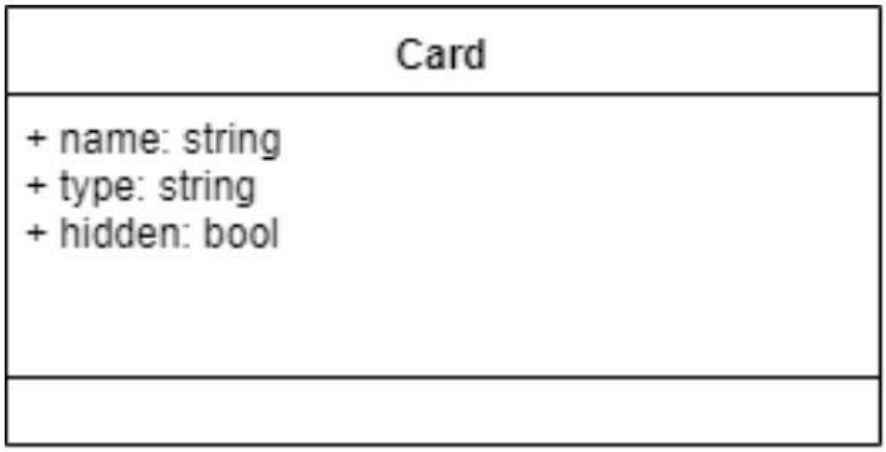
Module Coupling

This module doesn’t have a very high coupling because it simply displays the output of the process game state module. As long as the data coming into this module is presented in the way this module handles displaying the data, it should still function.

Design Organization

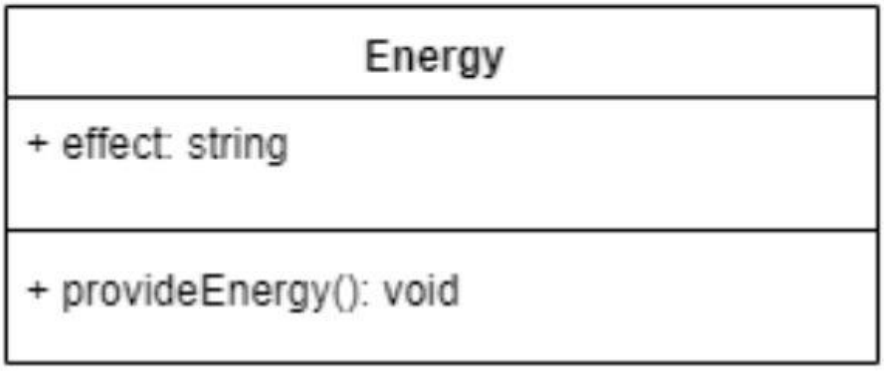
Detailed tabular description of Classes / Objects

Card Class



The card class is the base class that all cards will inherit from. All cards will have a string variable that holds the card's name, a string that holds the card's type, and a Boolean value that describes whether the card is known the player or not.

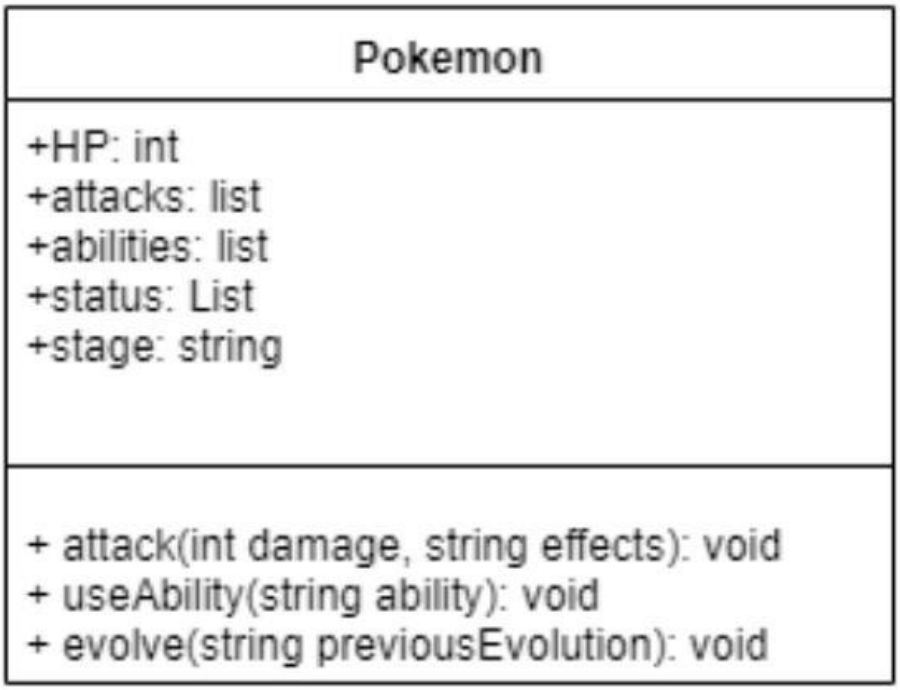
Energy Class



The Energy class will inherit from the Card class. It will have one string variable that will describe what kind of energy effect it has.

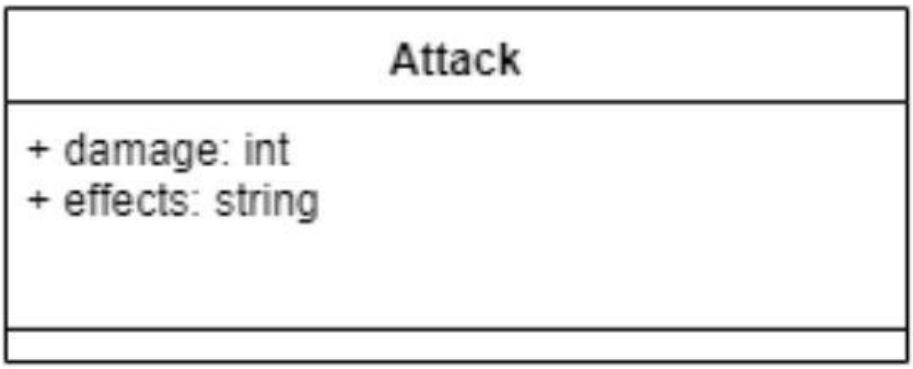
It will have one associate function that will provide energy to one Pokemon object in the Playfield class

Pokémon Class



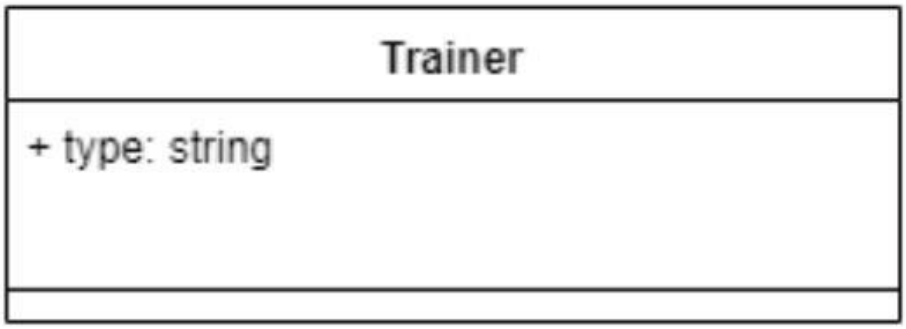
The Pokémon class will inherit from the Card class. It will have five additional attributes. These will describe the amount of health points (HP) it has as an integer, the attacks the Pokemon can perform as a list of Attack objects, the abilities the Pokemon can perform as a list of strings, any status aliments t he card has accrued as a list of strings, and the evolutionary stage of the Pokemon as a string.

Attack Class



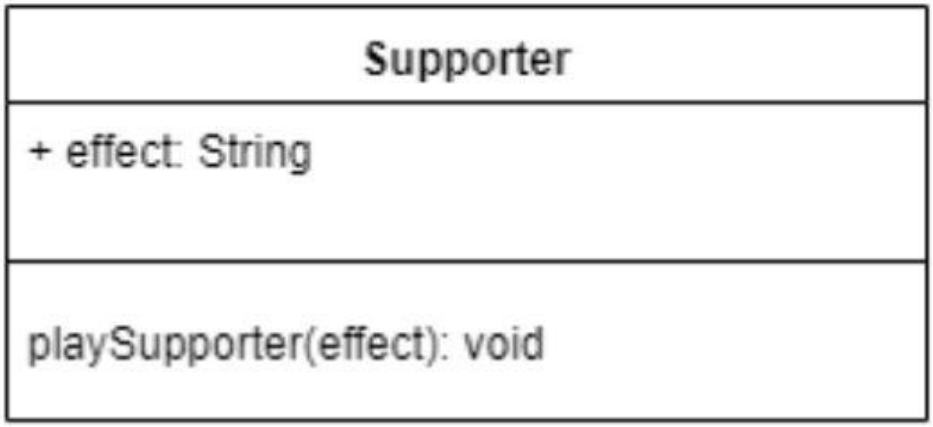
The Attack class will have a variable damage that will store an integer representing the amount of Health Points (HP) that will be deducted from the defending Pokemon. It will have a string variable describing any other effects the attack will cause.

Trainer Class



The Trainer class will inherit from the card class. It will have a single string type variable called type to help classify objects the inherit from it.

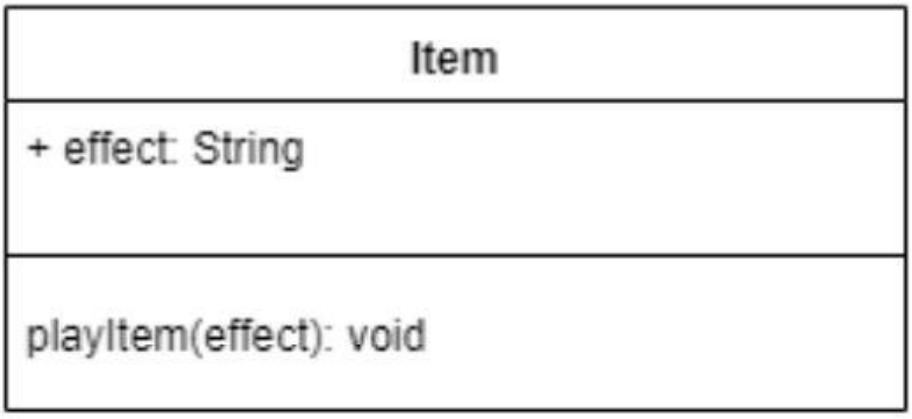
Supporter Class



The Supporter class will inherit from the Trainer class. It will have one additional attribute that will describe what kind of supporter effect it has stored as a string.

It will have one function that will play the card.

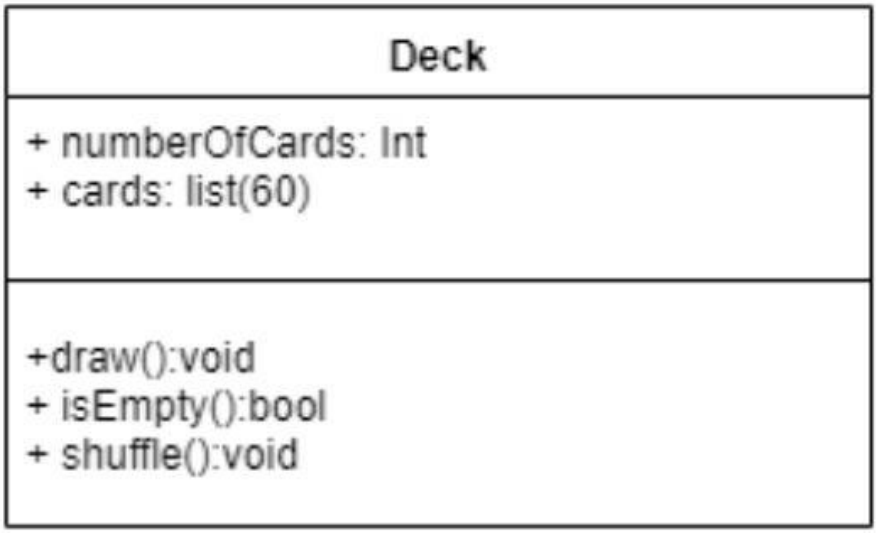
Item Class



The Item class will inherit from the Trainer class. It will have one additional attribute that will describe what kind of item effect it has as a string.

It will have one function that will play the card.

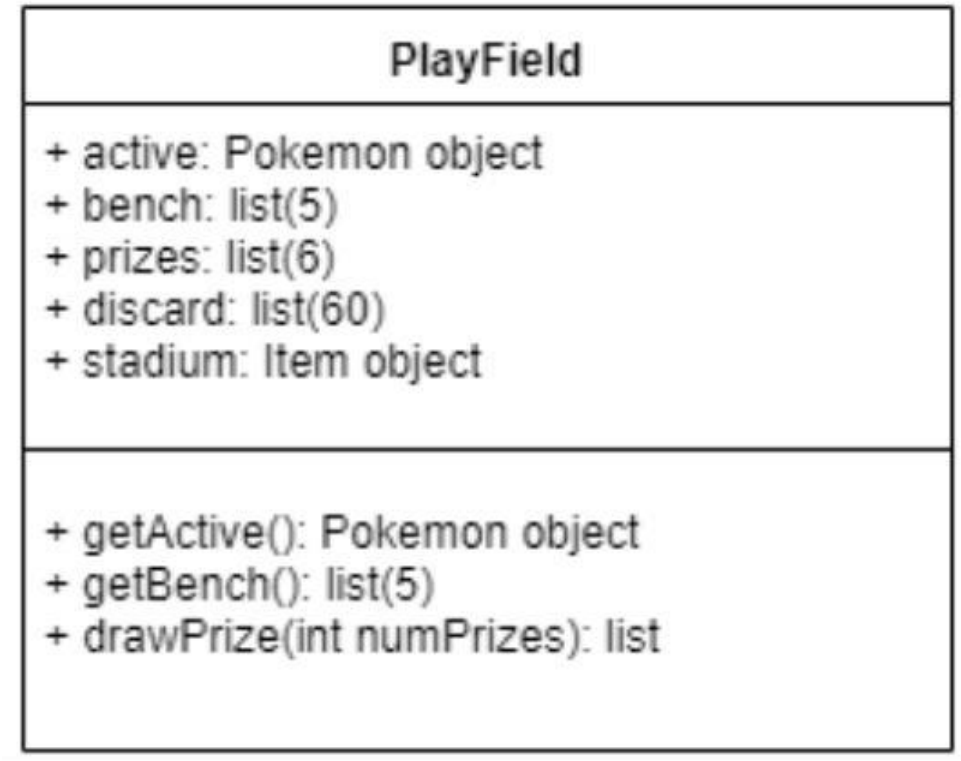
Deck Class



The Deck class will use an integer to keep track of the number of cards it contains. All of these cards will be stored in a list of Card objects.

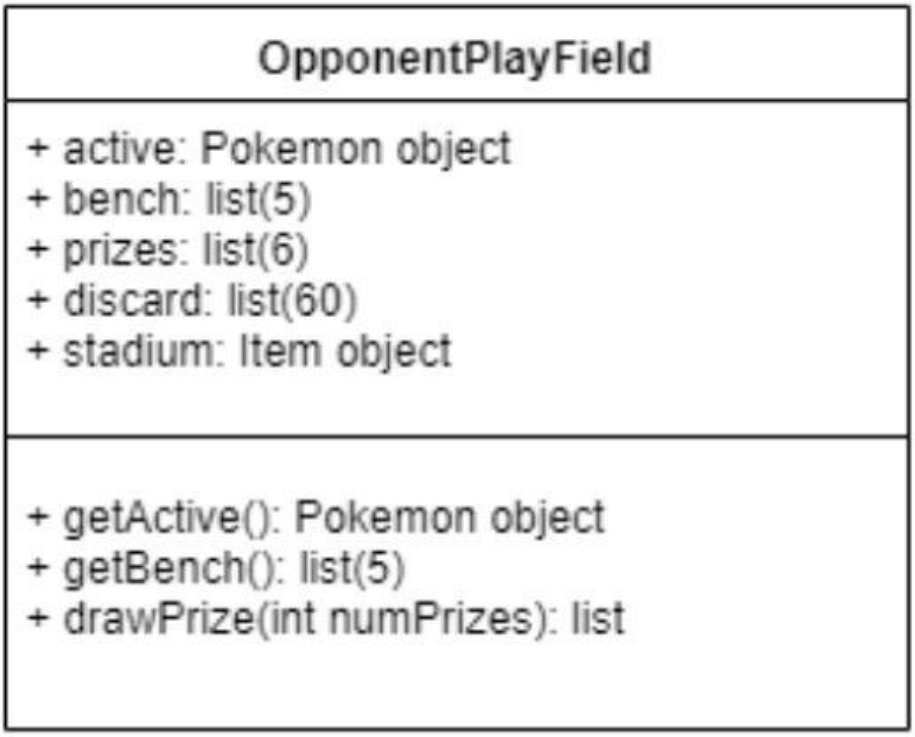
It will have 3 functions. The draw() function will simulate the drawing of a card by popping the first card off of the list and moving it to the Hand object. The isEmpty() function will return true if the cards list is empty, otherwise it will return false. The shuffle() function will rearrange the cards list to simulate shuffling a deck of cards.

Playfield Class



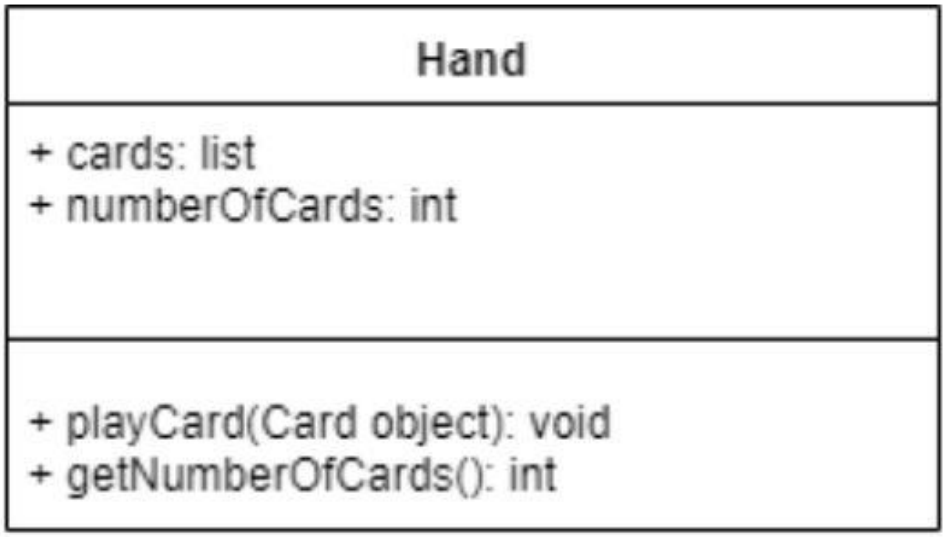
The playfield class will describe the game state. This includes all game zones controlled by the user. The active variable will hold a Pokemon object. This field keeps track of the player's Pokemon that is available to attack. The bench variable is a list of Pokemon objects. This represents the Pokemon the player has in play that are not available to attack. The prizes variable is a list of Card objects. These cards are used as a victory condition. The discard variable is a list of Card objects. The discard variable will have a maximum length of 60 because it needs to be equal to the Deck objects cards list as cards that are no longer in play are placed into this list to keep track of them. The stadium variable will contain an Item object.

OpponentPlayfield Class



The playfield class will describe the game state. This includes all game zones controlled by the opponent. The active variable will hold a Pokemon object. This field keeps track of the opponent's Pokemon that is available to attack. The bench variable is a list of Pokemon objects. This represents the Pokemon the opponent has in play that are not available to attack. The prizes variable is a list of Card objects. These cards are used as a victory condition. The discard variable is a list of Card objects. The discard variable will have a maximum length of 60 because it needs to be equal to the Deck objects cards list as cards that are no longer in play are placed into this list to keep track of them. The stadium variable will contain an Item object.

Hand Class



The Hand class will contain a list called cards consisting of Card objects. The numberOfCards variable will be an integer that holds the length of the cards list.

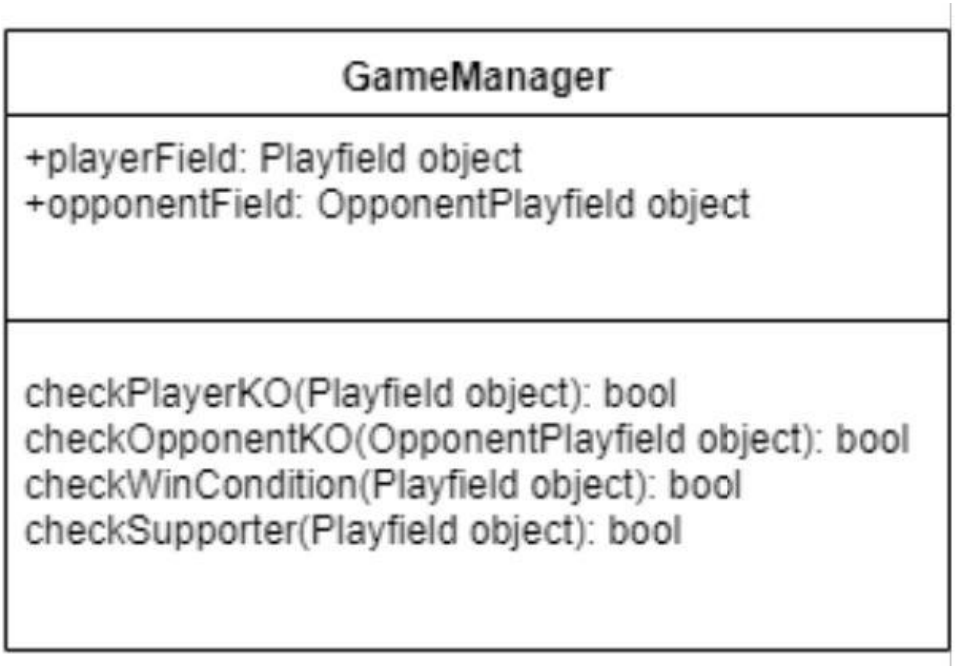
The playCard(Card object) function will move the card from the cards list and have any effects the card has sent to the proper module. The getNumberOfCards() function will simply return the value of the numberOfCards variable.

OpponentHand Class

The OpponentHand class will contain a numberOfCards variable will be an integer that will keep track of the number of cards in the opponent's hand.

The getNumberOfCards() function will simply return the value of the numberOfCards variable.

GameManager Class



The GameManger class will contain two variables. The playerField variable will contain a Playfield object. . The opponentField variable will contain an OpponentPlayfield object. This will give the GameManger object access to the entire game state.

It will contain four functions. The checkPlayerKO(Playfield object) will check the Playfield object's active variable which holds a Pokemon object to see if the HP variable has been reduced to less than or equal to zero. If so it will return true. The checkOpponentKO(OpponentPlayfield object) will check the OpponentPlayfield object's active variable which holds a Pokemon object to see if the HP variable has been reduced to less than or equal to zero. If so it will return true. The checkWinCondition(Playfield object) will determine if any win conditions have been met. If so it will return true. The checkSupporter(Playfield object)

will determine if a Supporter class object has been played on the current turn. If so it will return true. This is done because a player may play a maximum of one supporter card per turn.

Implementation Timeline

Implementation of “Project Vulpix” will begin during the spring 2019 semester. The project team consists of 4 students of California University of Pennsylvania, with current majors in Computer science. The team members are Christopher Crisson, Andrew Siddall, Matthew Bedillion, and Adlene Bellaoucha.

Andrew being the originator of the idea, and having knowledge of the programming software we chose, will be leading us through the implementation process. Christopher, Matthew, and Adlene, will also be working on programming under the direction of Andrew.

Our team has proposed a project that will require a large informational database, used to determine the output. Each input will trigger the project to delve into the database, searching for the appropriate response.

These functions are necessary in order to:

Determine the correct next step

Help the user learn to play the game more effectively

Determine effectiveness

- Will one Pokémon be effective against another

- Will a specific attack be effective against types

Determine energy needed for attacks

- Each attack needs a specific amount of energy of certain types to be performed

• Prompt for and read user input

• Verify appropriate inputs

• Display your best next move, clear and organized

In order for project Vulpix to run properly, the database must consist of all information going to be used in the game. Limitations to this would make certain cards unable to be recognized by the program. Many of the calculations performed are based on Pokémon types, these will be more of a comparison. The information the project consists of needs to be created and organized well, then the user will receive the correct output for what is given.

Design Testing

The developers of “Project Vulpix” have a desire to create a program that will be able to thoroughly teach the user how to play the Pokémon trading card game. With that in mind they will work hard on creating a database with the ability meet our needs. In order to do this, we will need to thoroughly test every aspect of the database within the program. This is necessary to make sure that the database is complete. Every piece of information from the database must be tested, that way we will know the information was entered correctly. With this testing we can assure that Project Vulpix will be able to provide clear outputs and avoid all crashes.

We will also want to be able to test through different access points. We will test access through the local point. It will be able to be used after someone has downloaded it. . We will test the program from different computers. Testing on computers with different operating systems will be useful to also insure compatibility.

Appendix: Technical Glossary

Ability: An Ability is an effect on a Pokémon that is not an attack. Some will be active all of the time, while some you will need to choose to use. Read each Ability to make sure you understand exactly how and when it works.

Active Pokémon: The player's in-play Pokémon that is not on the Bench. Only the Active Pokémon can attack.

Attach: When you take a card from your hand and put it on one of your Pokémon in play.

Attack: 1) When your Active Pokémon fights your opponent's Pokémon. 2) The text written on each Pokémon card that shows what it does when it attacks (a Pokémon can have several attacks on it).

Attacking Pokémon: The Active Pokémon, as it performs an attack.

Basic Energy Card: A Grass, Fire, Water, Lightning, Psychic, Fighting, Darkness, or Metal Energy card.

Basic Pokémon Card: A card you can play directly from your hand on your turn. *See Evolution card.*

Bench: The place for your Pokémon that are in play but are not actively fighting. They come out and fight if the Active Pokémon retreats or is Knocked Out. When Benched Pokémon take damage, do not apply Weakness or Resistance.

Burn marker: What you put on a Pokémon to remind you it is Burned. Remove the marker if the Pokémon is Benched or Evolved. *See counter, damage counter.*

Counter: Object put on a Pokémon as a reminder (for example, a Char counter). A counter does not go away when you Bench the Pokémon, but it does go away if the Pokémon evolves (damage counters are a special exception to this rule). *See damage counter, Poison marker, Burn marker.*

Damage: What usually happens when one Pokémon attacks another. If a Pokémon has total damage greater than or equal to its Hit Points, it is Knocked Out.

Damage Counter: A counter put on your Pokémon to show it has taken damage. It stays on your Pokémon even if the Pokémon is Benched or evolved. Each damage counter counts as much damage as it has printed on it. *See counter, Poison marker.*

Defending Pokémon: Your Active Pokémon when your opponent is attacking.

Devolve: Certain cards can devolve an Evolved Pokémon, which is the opposite of evolving your Pokémon. When a Pokémon is devolved, it also loses Special Conditions and any other effects.

Discard Pile: The cards you have discarded. These cards are always face up. Anyone can look at these cards at any time.

Energy Card: Cards that power your Pokémon so they can attack. *See basic Energy card.*

Evolution Card: A card you play on top of a Basic Pokémon card (or on top of another Evolution card) to make it stronger.

Fossil Trainer cards: A special kind of Trainer card that acts like a Basic Pokémon when put into play. When a Fossil Trainer card is in your hand, deck, or discard pile, it is not considered a Basic Pokémon. However, these Trainer cards always count as Basic Pokémon during set-up.

Hit Points (HP): A number every Pokémon has, telling you how much damage it can take before it is Knocked Out.

In-Between Turns: The part of each turn when the game shifts from one player to the other. Check Poison, Burn, Asleep, and Paralysis at this step, and see whether any Pokémon are Knocked Out.

In Play: Your cards are in play when they are on the table. Basic Pokémon cards, Evolution cards, and Energy cards cannot be used unless they are in play. (The cards in your deck, your discard pile, and your Prizes are not in play, but your Benched Pokémon are.)

Knocked Out: A Pokémon is Knocked Out if it has damage greater than or equal to its Hit Points. That Pokémon goes to the discard pile along with all cards attached to it. When one of your opponent's Pokémon is Knocked Out, take one of your Prizes.

Item card: A type of Trainer card. Follow the instructions on the card and then discard it.

Lost Zone: Cards sent to the Lost Zone are no longer playable during that match. Put them face up anywhere out of play.

Owner: A Pokémon with a Trainer's name in its title, such as Brock's Sandshrew or Team Rocket's Meowth.

Poison Marker: Object put on a Pokémon to remind you it is Poisoned. Remove the marker if the Pokémon is Benched or evolved. *See counter, damage counter.*

Poké-Body: An effect that is active as soon as that Pokémon card is in play and lasts until the Pokémon leaves play.

Poké-Power: A once-per-turn power on Active and Benched Pokémon you must choose to use. Most Poké-Powers are turned off if the Pokémon has a Special Condition.

Pokémon: The colorful characters that fight for you in the Pokémon Trading Card Game. They are represented in the game by Basic Pokémon and Evolution cards.

Pokémon-ex: Pokémon-ex are a stronger form of Pokémon with a special drawback: when your Pokémon-ex is Knocked Out, your opponent draws two Prize cards instead of one.

Pokémon LEGEND: Special double cards that showcase powerful Legendary Pokémon. Both cards must be played together at the same time.

Pokémon LV.X: Stronger versions of a regular Pokémon, put on top of the regular Pokémon of the same name and adding extra abilities to the original Pokémon.

Pokémon Power: A special ability some Pokémon have. Pokémon Powers are divided into two categories: Poké-Power and Poké-Body. They always include the words “Poké-Power” or “Poké-Body” so you can tell they are not attacks.

Pokémon SP: A special Pokémon trained by a particular Trainer, with a symbol in its name to show its owner.

Pokémon Tool: A special kind of Trainer card (an Item) you can attach to your Pokémon to help you. Each Pokémon can have only 1 Pokémon Tool attached at any time.

Prize Cards: The 6 cards you put face down at the start of the game. Every time one of your opponent's Pokémon is Knocked Out, you take 1 of your Prizes into your hand (or 2 Prizes, for a Pokémon-ex). When you take your last Prize card, you win!

Resistance: A Pokémon with Resistance takes less damage when attacked by Pokémon of a certain type. The amount of Resistance is printed next to the type of Resistance(s) a Pokémon has, if any.

Retreat: When you switch your Active Pokémon with one of your Benched Pokémon. To retreat, you must discard Energy from the retreating Pokémon equal to the Retreat Cost of the Pokémon. This cost appears in the lower right-hand corner of the card. You can only retreat once per turn.

Special Conditions: Asleep, Burned, Confused, Paralyzed, and Poisoned are called Special Conditions.

Stadium Card: A type of Trainer card similar to an Item card, but stays in play after you play it. Only one Stadium card can be in play at a time—if a new one comes into play, discard the old one and end its effects. You can play only one Stadium card each turn.

Sudden Death: Sometimes both players win at the same time. In this case, you play a short game called "Sudden Death" (use only 1 Prize card each instead of 6).

Supporter Card: A Trainer card similar to an Item card. You can play only one Supporter card each turn.

Technical Machine: A kind of Trainer card (an Item) you can attach to your Pokémon. When attached, your Pokémon can use the Technical Machine attack as its own. Technical Machine cards remain attached unless the card text says otherwise.

Trainer Card: Special cards you play to gain advantages in the game. *See Item card, Stadium card, Supporter card.*

Trainers' Pokémon: Pokémon with Trainers' names in their titles, like Brock's Sandshrew. You cannot evolve a regular Sandshrew into Brock's Sandslash, and you cannot evolve a Brock's Sandshrew into a regular Sandslash. This is because "Brock's" is part of the name.

Weakness: A Pokémon with Weakness takes more damage when attacked by Pokémon of a certain type. The effect of the Weakness is indicated next to the type(s) of Weakness a Pokémon has, if any.

Appendix: Team Details

Mathew Bedillion was primarily responsible for the Implementation Timeline, Design Testing and Project Block Diagram sections.

Adlene Bellaoucha was primarily responsible for the Abstract, Description of the Document and Writing Center Report sections.

Andrew Siddall was primarily responsible for the System Modules and Resposibilities sections.

Christopher Crisson was primarily responsible for the Design Organization and Team Details sections.

All group members supported the others with there sections by communicating effectively and editing each other’s work throughout the process of creating this document.

Appendix: Writing Center report

Client: Adlene Bellaoucha

Staff or Resource: James R.

Date: December 6, 2018, 5:00pm - 6:00pm

Did the student request that the instructor receive a visit report?: Yes

What course was serviced by this visit?: CSC490

What goals were established for this tutoring session?: Organizing; Revising; Editing;

How did the process of this consulting session address the established goals?: The writing center consultant offered a number of edits focused on helping the authors achieve more concision and clarity in their writing. In addition, a few comments on format were offered. Finally, a number of questions were asked, since this document is aimed to members of an academic community (computer science) who might not fully understand the game that is the focus of this software design; we wanted to challenge the author

Please provide any additional comments relevant to this session.: This project sounds really interesting. We wish you great success as you move forward!

Appendix: Workflow Authentication

I, Andrew Siddall, hereby declare and confirm that I have performed the work as documented herein.

Signature: _____

Date: _____

I, Adlene Bellaoucha, hereby declare and confirm that I have performed the work as documented herein.

Signature: _____

Date: _____

I, Mathew Bedillion, hereby declare and confirm that I have performed the work as documented herein.

Signature: _____

Date: _____

I, Christopher Crisson, hereby declare and confirm that I have performed the work as documented herein.

Signature: _____

Date: _____