AP Computer Science Final Project - README Template

Instructions:
The first step in creating an excellent APCS final project is to write up a README. At this stage, this README file acts as your **project proposal**. Once you've filled in all components, Shelby will read through it and suggest edits. Ultimately, you need a document that adequately describes your project idea and **we must agree on this plan**.

Have one member of your group **make a copy of this Google Doc**. Then, they should share it with all other members **and with Mr. Shelby** so that every group member has edit permissions, and Shelby can add comments on your ideas.

There's a lot of parts of this document that you might not have full answers for yet. Because you haven't written the program yet, it's difficult to think about the **instructions** or **which group members will do which parts**. Even though this is hard to think about, you must have something in these sections that acts as your current plan. However, during the course of the project, you'll **continuously update this document**. This means that you will not be *held* to exactly what you put here - components of this document can change (and it's pretty common!).

There is one exception: the **Features List** section. Once Shelby OKs your README, the Features List section **cannot be modified**. For this reason, it is most important that you get a solid idea of what you want to make and the primary features it will have *now*.

Talk with your group. Consider drawing some pictures of what you think your project might look like. Be precise. When you're ready, fill this out together. Each component in brackets below ([these things]) should be replaced with your ideas. Note that there are several sample READMEs posted on this assignment for you to use as guidance.

-------------------When README is finalized, remove everything above this line--------------------

Warriors

Authors: Ishaan Musunuri, Frank An
Revision: 5/06/22

Introduction:
[In a few paragraphs totaling about ½ page, introduce the high-level concept of your program. What this looks like depends a lot on what type of thing you are making. An introduction for an *application* will look different than one for a *game*. In general, your introduction should address questions like these:
What does your program do?

What problem does it solve? Why did you write it?
What is the story?
What are the rules? What is the goal?
Who would want to use your program?
What are the primary features of your program?]

Our program runs a battle game where two players fight to see who the better fighter is. Solve the mystery of the chosen one who is destined to win the battle. See the breathtaking game written by Frank An and Ishaan Musunuri and continue on this adventure to be the greatest fighter among your friends and other players as well. The rules of the game

<u>Instructions</u>:
[Explain how to use the program. This needs to be **specific**:
Which keyboard keys will do what?
Where will you need to click?
Will you have menus that need to be navigated? What will they look like?
Do actions need to be taken in a certain order?]

Players will have the option to look at the instructions, or to begin playing the game.
Players must choose two characters from the starting screen. Different characters will have different abilities such as attack power, the type of special move, and the speed.
After players choose, they will be transported to a battlefield where they will use their chosen characters to fight each other .
Goal of the game is to defeat your opponent.
Players will be able to punch, and use a special move. The special move will be limited.
Once there is a winner, there will be a pop up stating the winner, and giving the players the opportunity to play another game.
<u>Controls:</u>

- Player 1:
    - W-move up
    - A - move back
    - D - move forward
    - S - move down
    - X - special move (limited uses)
- Player 2:
    - Up - move up
    - Down - move down
    - Left - move back
    - Right - move forward

<u>Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER)</u>:

Must-have Features:
[These are features that we agree you will *definitely* have by the project due date. A good final project would have all of these completed. At least 5 are required. Each feature should be fully described (at least a few full sentences for each)]

-
-
-
-
-

Want-to-have Features:
[These are features that you would like to have by the project due date, but you're unsure whether you'll hit all of them. A good final project would have perhaps half of these completed. At least 5 are required. Again, fully describe each.]

-
-
-
-
-

Stretch Features:
[These are features that we agree a *fully complete version of this program would have, but that you probably will not have time to implement*. A good final project does not necessarily need to have any of these completed at all. At least 3 are required. Again, fully describe each.]

-
-
-

Class List:
[This section lists the Java classes that make up the program and very briefly describes what each represents. It's totally fine to put this section in list format and not to use full sentences.]

- Main
- Weapons
- DrawingSurface
- Character
    - Bob
    - Steve
    - Jeff
    - Carl
    - Dave
- Map
-

<u>Credits</u>:
[Gives credit for project components. This includes both internal credit (your group members) and external credit (other people, websites, libraries). To do this:

- Ishaan - UML, README
- Frank - README
- Give credit to all outside resources used. This includes downloaded images or sounds, external java libraries, parent/tutor/student coding help, etc.]