NEA Celebrity Dogs User Requirements

# User Need 1

## A Menu

A user-friendly menu must be included in the program. It will need to have at least 2 options, with at least 3 outcomes. This will require the user to input a string, the way the choice will be presented is alongside numbers the user will input to finally make a choice. This is to make the program as a whole more user-friendly, so the user finds the program easy and preferable to use.

* Option & Outcome 1 is the user choosing to play the program. If this is chosen, the relevant subroutines will have to be ran.
* Option & Outcome 2 is a quit functionality.
  + By using asking the user for a Boolean input and utilising IF & ELSE, the program will make sure the user wants to quit and will return them to the program in the case of any answer that isn’t *“yes”*
* Outcome 3 will handle the code ran if the user inputs a valid command. It will tell the user they have entered an invalid command and will call the *menu* subroutine again, effectively functioning as a loop. This is the best way to do this, as it allows me to call on the menu at any point during the code, allowing for an easily built round system or error handler.

# User Need 2

## User Readability and Card Management

The Program will need to display the top card of the user’s deck. The way the decks are handled will be identically to the CPU deck management, the only different being the program will read the cards name and its statistics, and then print them in a readable format. This is to promote longer user play-times and allow the user to actually play the game, so they know what card is on the top of their deck.

* This format could be printed using ASCII Characters, to present the card in a format similar to an actual card.
* This can be achieved with simple printing, however, I will need to reference a 2D array for different pieces of data from each card.
* This is optimal, because referencing a 2D array would require less variable declarations than if I defined each card as a tuple, making a 2D array the most efficient solution.
* In the case of an array not having an assigned value, I will have my program “abandon” that card, and attempt to swap that card with another card.

# User Need 3

## The Ability to choose a category

After asking the user what category they would like to choose, the user will input a string, the program will confirm the right category is chosen by prompting the user with a Boolean yes or no question. This is needed so the game can be played properly to the degree specified in the NEA.

* This can be achieved using a *userinput*, firstly comparing it to the values exercise, friendliness, intelligence or drool. Then, I will need to ask the user if they are sure they want to choose this category.
  + In order to execute this in my code, I will need to utilise comparative if and else if statements. There will also need to be an *else* statement in order to call the subroutine again if the user makes a mistake or enters an invalid input.

# User Need 4

## A Win or Loss Message

The Program will need to display the relevant message upon the end of a round. This will need to be display in a noticeable manor and in a way the user can’t easily overlook.

* What message is displayed will be derived from the comparison made by either card in the previous round.
* The comparison will be achieved with a simple if, else if & else statement
  + The first comparison will be an *if* statement, and will print the message and perform other required tasks for a scenario where the user *wins* the round.
  + The second comparison will be an *else if* statement, and will print and perform when theplayer *loses* the round.
  + The third and final “fall-back” code will be in the case of a draw, in which “Draw” will be printed and the round will repeat.
* There is no predictable scenario in which there will be an error, so I believe that no error handling will be needed.

# User Need 5

## Deciding on how many rounds are to be played

At the beginning of a session, a user will need to decide how many rounds they wish to play. This will need to be an even *whole* integer. This will then set a variable which will cause a *loop* until the round is complete. This is so the user can choose how many rounds are appropriate for them at this time, so they aren’t forced to consistently repeat the game or are forced to play for too long.

* Utilising the *try* module in Python, I can make sure what the user inputs is a whole integer, if not the user will be informed and prompted again for input.
  + The exact code line will be “*try* int(userinput):”
* The loop will centre around a variable called rounds and use the line “for round in rounds:” to run the game’s main functional subroutine

# User Need 6

## Finalizing how many cards are to be in a Deck

The User will need to decide how many cards are going to be played in total, this will need to be an amount that is a whole integer and divisible by two. This means the programming will be similar to User Need 5, with some minor differences.

* Again utilising the python try module, the users input will need to be checked against two try statements, in order to ensure that user has entered a valid integer divisible by two.
  + These try snippets are:  
    “try int(userinput)”  
    “try int(userinput/2)”
* If a value error occurs, I will use nested subroutines in order to perform as a loop, the program will call its own subroutine after printing an error message. The loop will cause the program to prompt the user for input again.