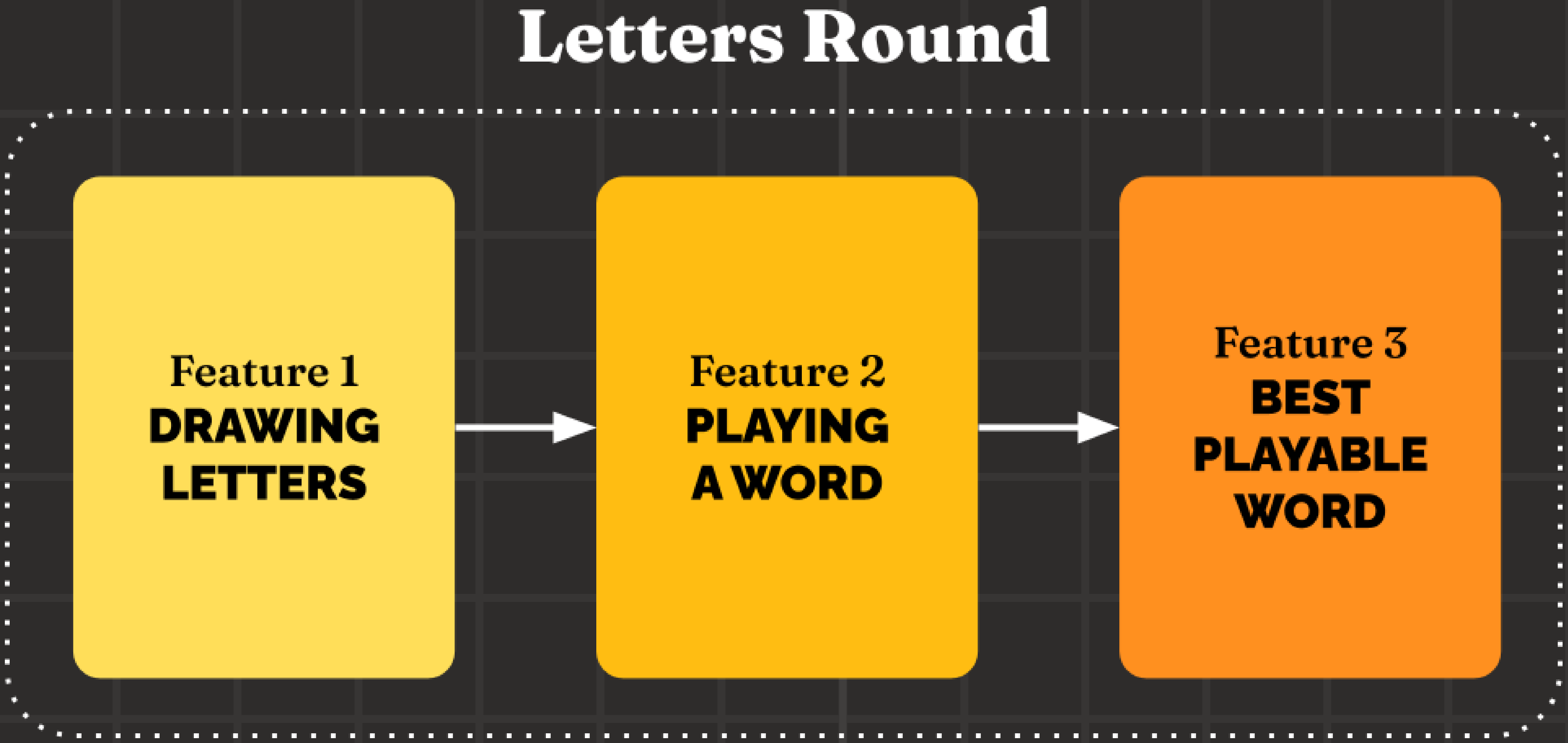
Sanjeev.

TERMINAL APP WALKTHROUGH

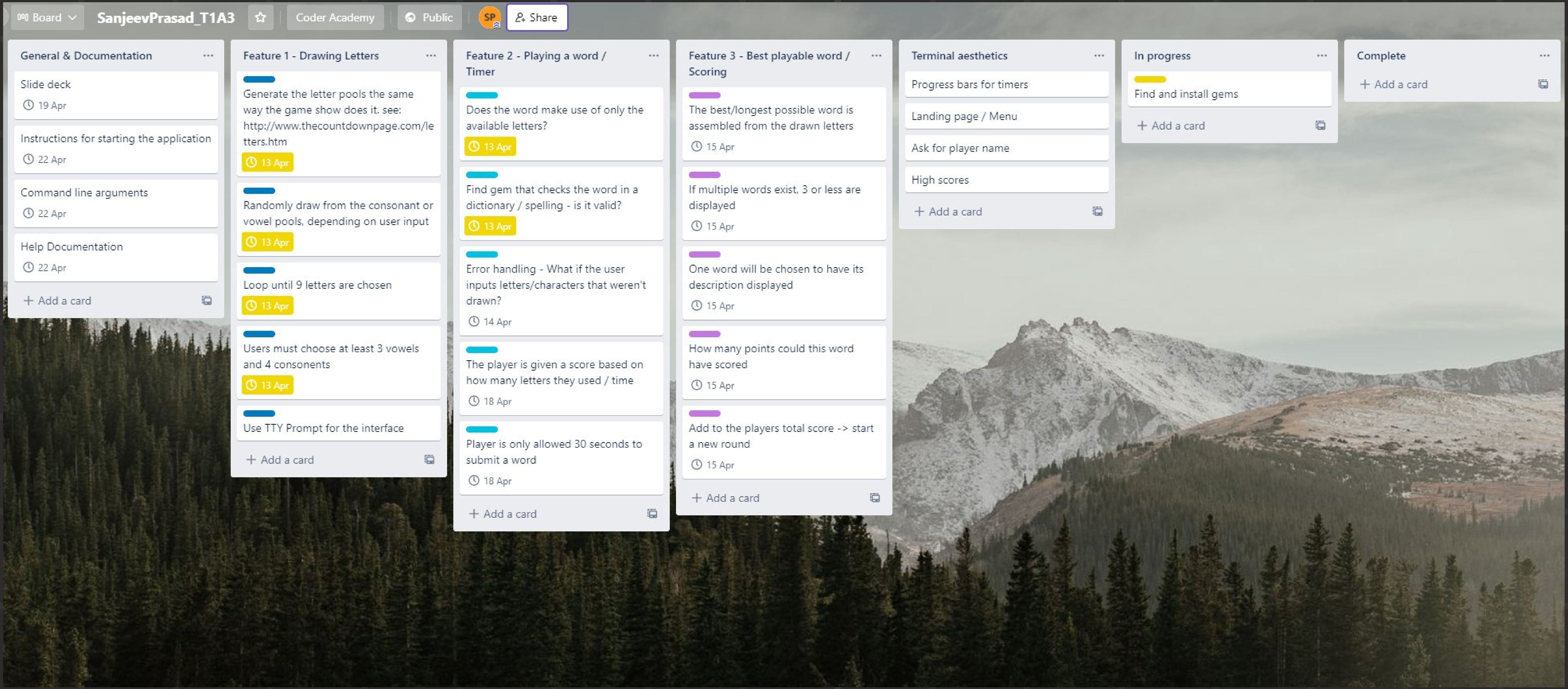
The Idea.

- **Countdown** game show
- The application will recreate the **Letters round** from the show
- Possibility of adding numbers round and conundrum if time permits.

Overall Structure.



Implementation.



General & Documentation

- Slide deck — 19 Apr
- Instructions for starting the application — 22 Apr
- Command line arguments — 22 Apr
- Help Documentation — 22 Apr
- + Add a card

Feature 1 - Drawing Letters

- Generate the letter pools the same way the game show does it. see: http://www.thecountdownpage.com/letters.htm — 13 Apr
- Randomly draw from the consonant or vowel pools, depending on user input — 13 Apr
- Loop until 9 letters are chosen — 13 Apr
- Users must choose at least 3 vowels and 4 consonents — 13 Apr
- Use TTY Prompt for the interface
- + Add a card

Feature 2 - Playing a word / Timer

- Does the word make use of only the available letters? — 13 Apr
- Find gem that checks the word in a dictionary / spelling - is it valid? — 13 Apr
- Error handling - What if the user inputs letters/characters that weren't drawn? — 14 Apr
- The player is given a score based on how many letters they used / time — 18 Apr
- Player is only allowed 30 seconds to submit a word — 18 Apr
- + Add a card

Feature 3 - Best playable word / Scoring

- The best/longest possible word is assembled from the drawn letters — 15 Apr
- If multiple words exist, 3 or less are displayed — 15 Apr
- One word will be chosen to have its description displayed — 15 Apr
- How many points could this word have scored — 15 Apr
- Add to the players total score -> start a new round — 15 Apr
- + Add a card

Terminal aesthetics

- Progress bars for timers
- Landing page / Menu
- Ask for player name
- High scores
- + Add a card

In progress

- Find and install gems
- + Add a card

Complete

- + Add a card

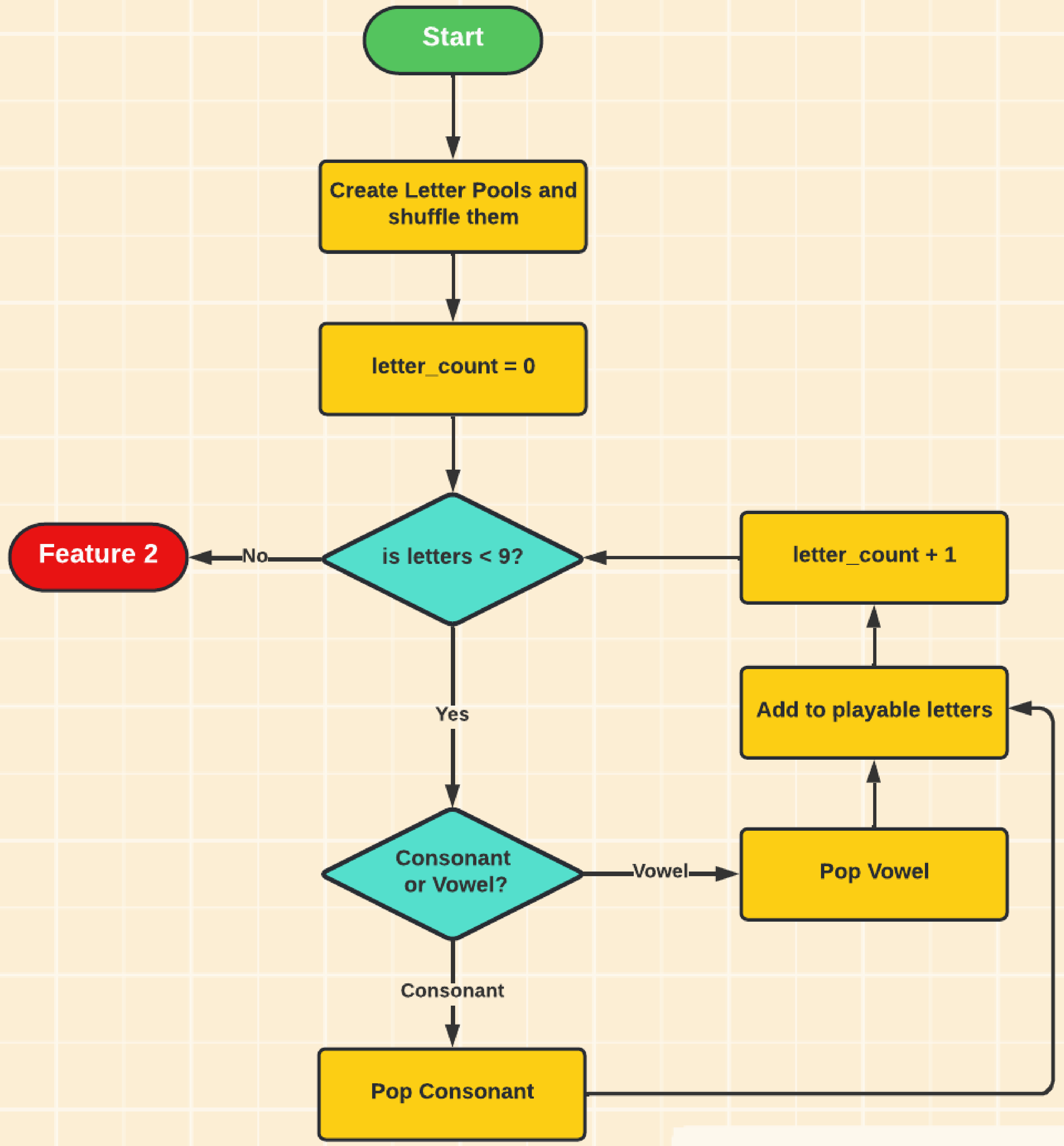
Feature 1.

DRAWING LETTERS

- Generate letter pools using the same frequency of letters as the game show
- Players can choose a **vowel** or **consonant**
- Loop until **9 letters** are chosen
- Users must choose at least **3 vowels** and **4 consonants**
- Interface built using TTY Prompt

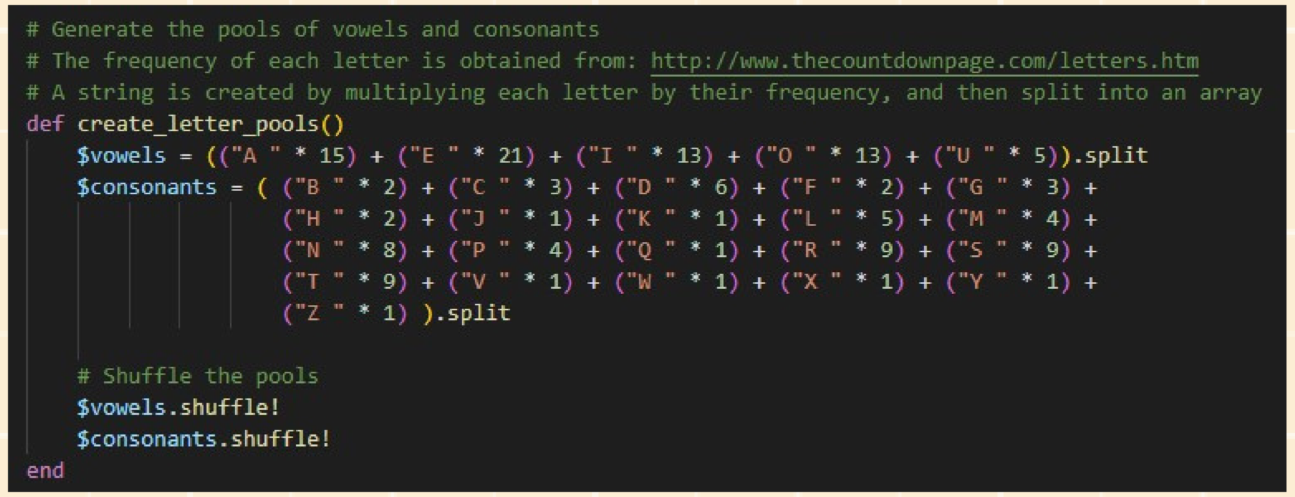
ERROR HANDLING

- What if the user chooses too many vowels/consonants?



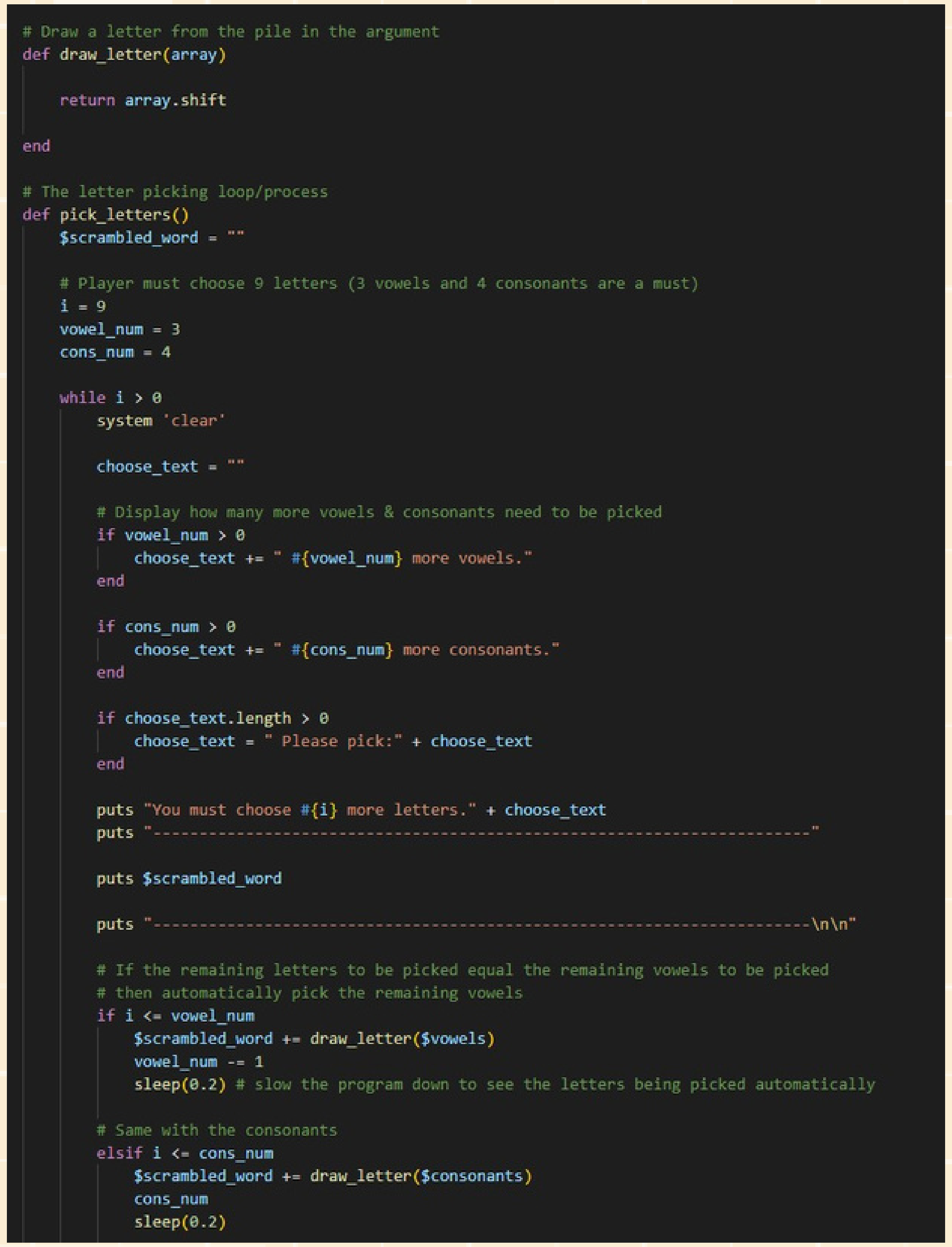
Feature 1.

GENERATING LETTER POOLS

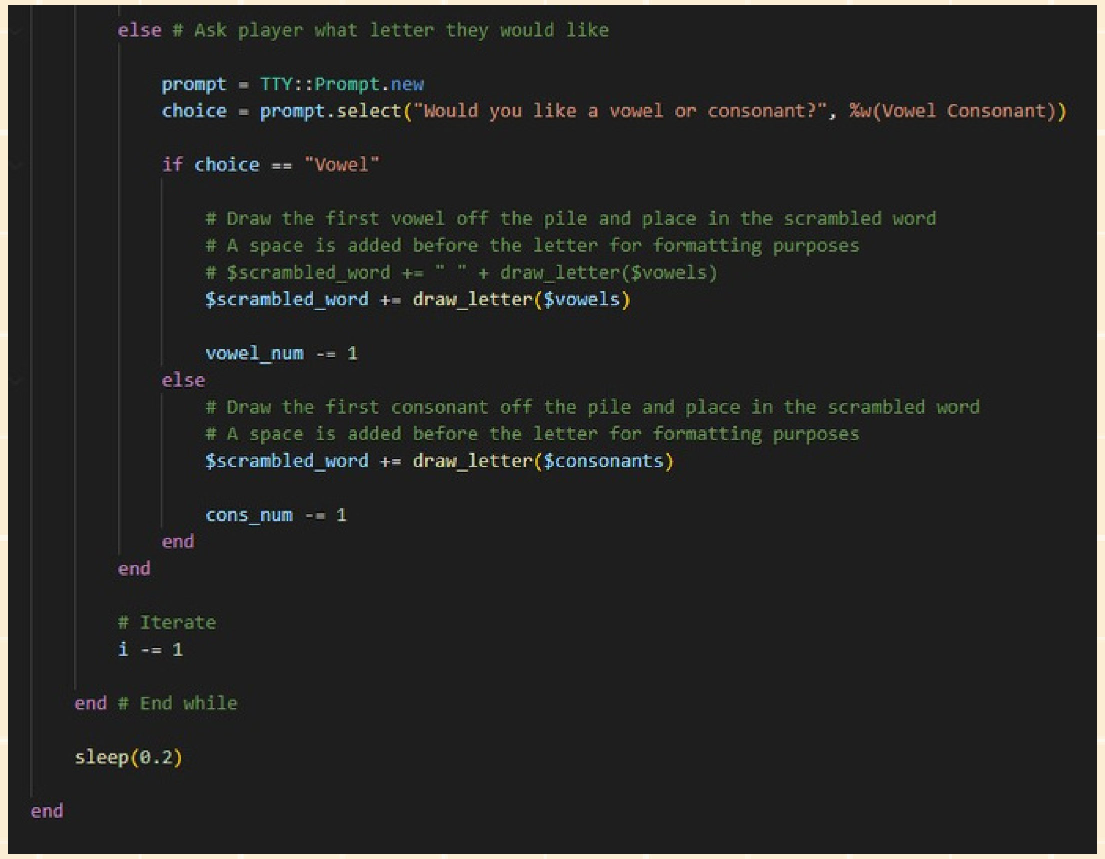


```ruby
# Generate the pools of vowels and consonants
# The frequency of each letter is obtained from: http://www.thecountdownpage.com/letters.htm
# A string is created by multiplying each letter by their frequency, and then split into an array
def create_letter_pools()
    $vowels = (("A " * 15) + ("E " * 21) + ("I " * 13) + ("O " * 13) + ("U " * 5)).split
    $consonants = ( ("B " * 2) + ("C " * 3) + ("D " * 6) + ("F " * 2) + ("G " * 3) +
                    ("H " * 2) + ("J " * 1) + ("K " * 1) + ("L " * 5) + ("M " * 4) +
                    ("N " * 8) + ("P " * 4) + ("Q " * 1) + ("R " * 9) + ("S " * 9) +
                    ("T " * 9) + ("V " * 1) + ("W " * 1) + ("X " * 1) + ("Y " * 1) +
                    ("Z " * 1) ).split

    # Shuffle the pools
    $vowels.shuffle!
    $consonants.shuffle!
end
```

DRAWING LETTERS



```ruby
# Draw a letter from the pile in the argument
def draw_letter(array)

    return array.shift

end

# The letter picking loop/process
def pick_letters()
    $scrambled_word = ""

    # Player must choose 9 letters (3 vowels and 4 consonants are a must)
    i = 9
    vowel_num = 3
    cons_num = 4

    while i > 0
        system 'clear'

        choose_text = ""

        # Display how many more vowels & consonants need to be picked
        if vowel_num > 0
            choose_text += " #{vowel_num} more vowels."
        end

        if cons_num > 0
            choose_text += " #{cons_num} more consonants."
        end

        if choose_text.length > 0
            choose_text = " Please pick:" + choose_text
        end

        puts "You must choose #{i} more letters." + choose_text
        puts "------------------------------------------------------------------------"

        puts $scrambled_word

        puts "----------------------------------------------------------------------\n\n"

        # If the remaining letters to be picked equal the remaining vowels to be picked
        # then automatically pick the remaining vowels
        if i <= vowel_num
            $scrambled_word += draw_letter($vowels)
            vowel_num -= 1
            sleep(0.2) # slow the program down to see the letters being picked automatically

        # Same with the consonants
        elsif i <= cons_num
            $scrambled_word += draw_letter($consonants)
            cons_num
            sleep(0.2)
```



```ruby
        else # Ask player what letter they would like

            prompt = TTY::Prompt.new
            choice = prompt.select("Would you like a vowel or consonant?", %w(Vowel Consonant))

            if choice == "Vowel"

                # Draw the first vowel off the pile and place in the scrambled word
                # A space is added before the letter for formatting purposes
                # $scrambled_word += " " + draw_letter($vowels)
                $scrambled_word += draw_letter($vowels)

                vowel_num -= 1
            else
                # Draw the first consonant off the pile and place in the scrambled word
                # A space is added before the letter for formatting purposes
                $scrambled_word += draw_letter($consonants)

                cons_num -= 1
            end
        end

        # Iterate
        i -= 1

    end # End while

    sleep(0.2)

end
```

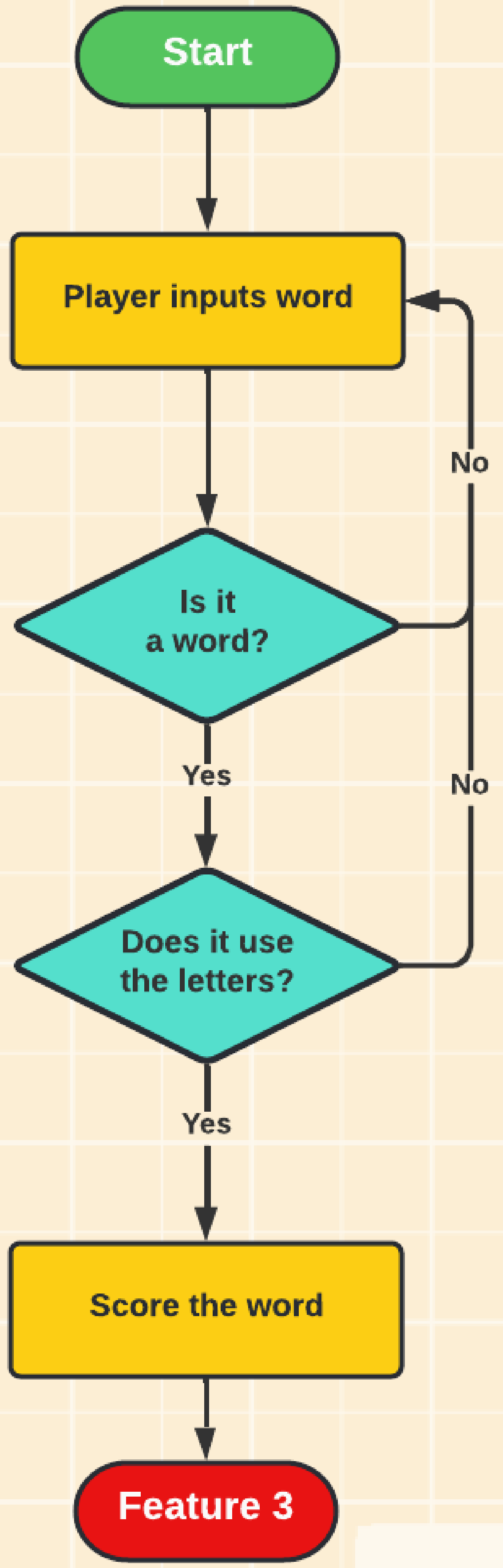
Feature 2.

PLAYING A WORD

- Allow user to play a word
- Check validity of the word using a dictionary gem
- Did the player use only the available letters?
- Give the description of the played word
- Possibility of adding a time limit later on

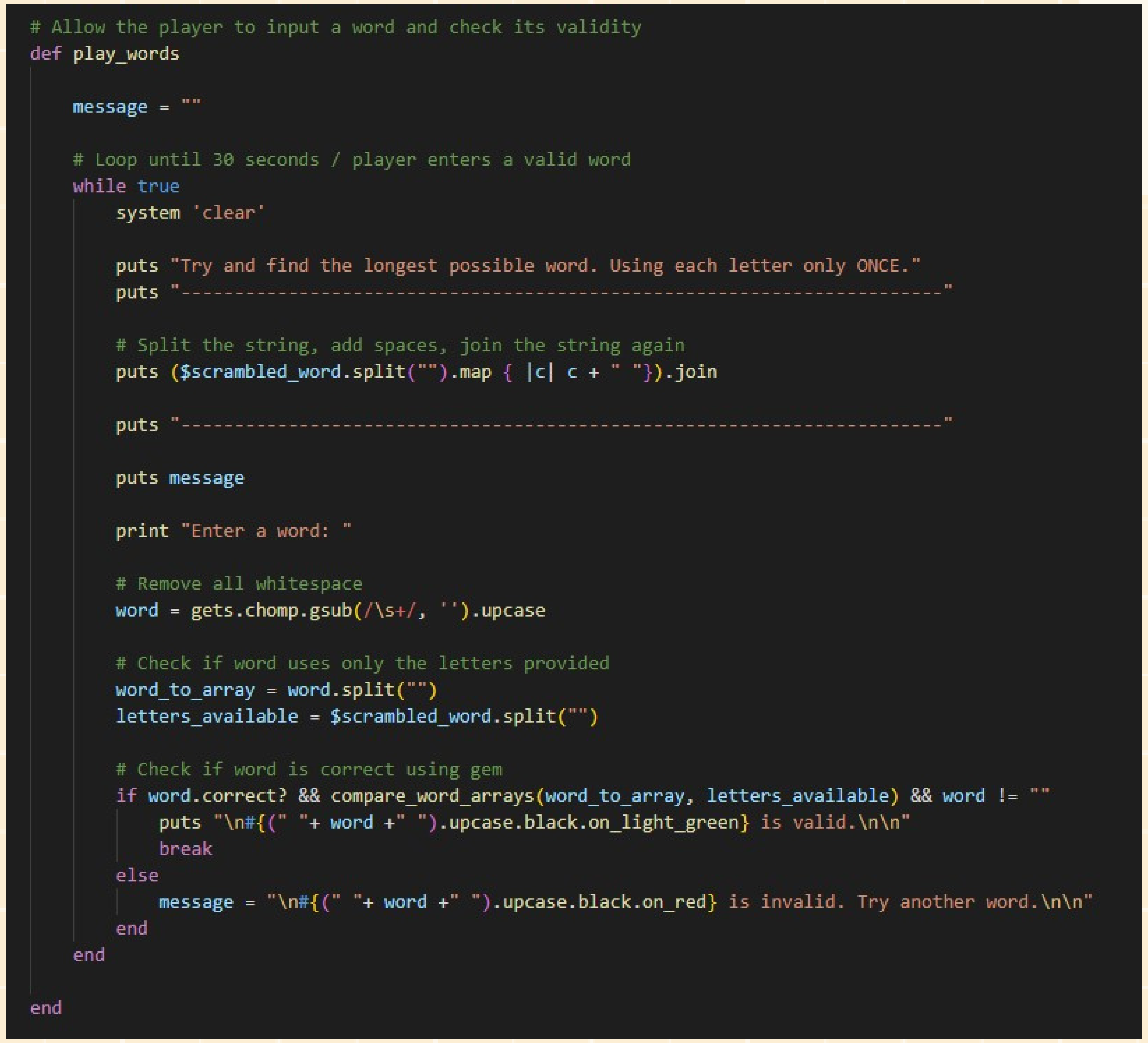
ERROR HANDLING

- What if the player enters invalid characters?
- What if the same letter is used twice?
- Network errors (dictionary gem)



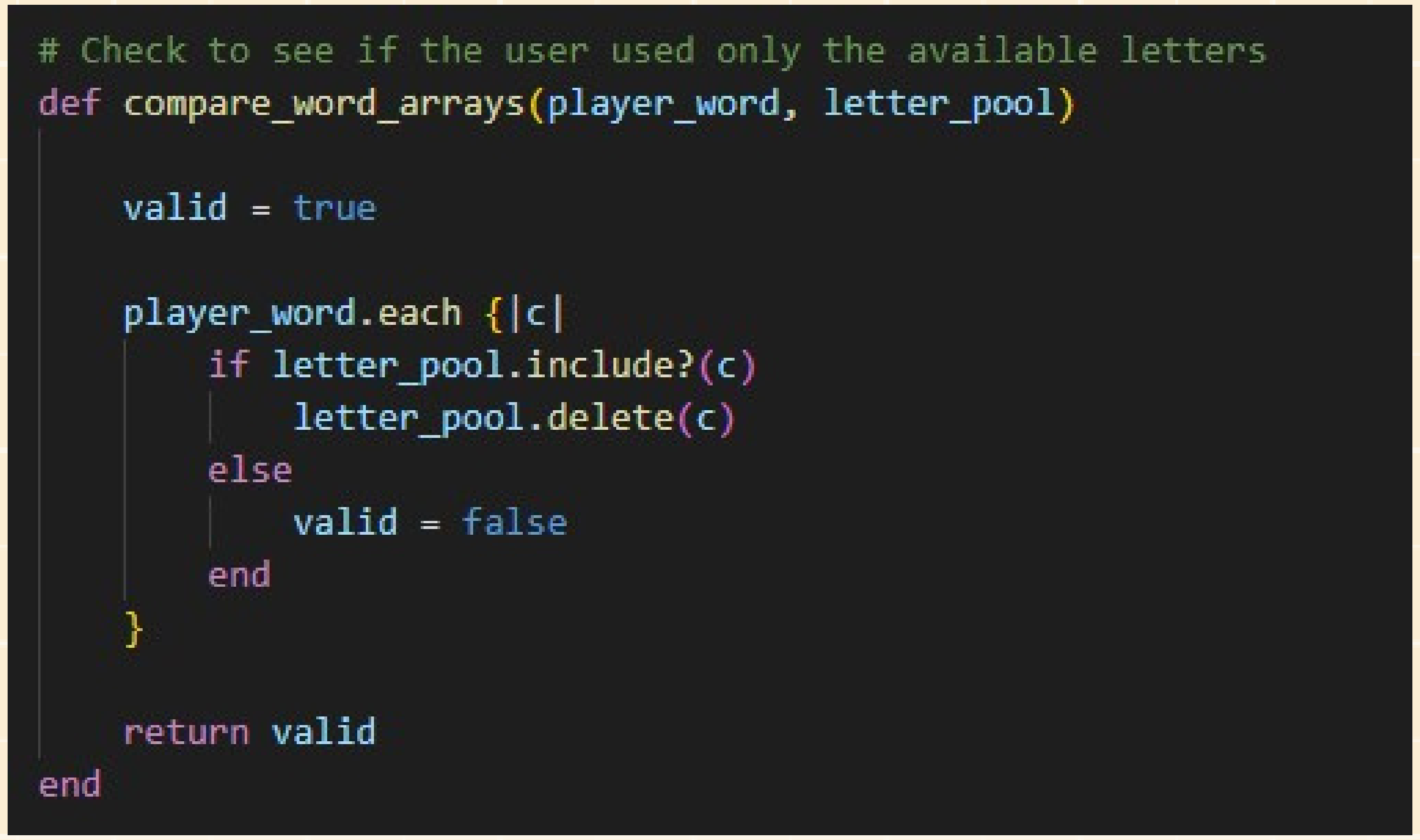
Feature 2.

PLAY WORDS



```ruby
# Allow the player to input a word and check its validity
def play_words

    message = ""

    # Loop until 30 seconds / player enters a valid word
    while true
        system 'clear'

        puts "Try and find the longest possible word. Using each letter only ONCE."
        puts "-----------------------------------------------------------"

        # Split the string, add spaces, join the string again
        puts ($scrambled_word.split("").map { |c| c + " "}).join

        puts "-----------------------------------------------------------"

        puts message

        print "Enter a word: "

        # Remove all whitespace
        word = gets.chomp.gsub(/\s+/, '').upcase

        # Check if word uses only the letters provided
        word_to_array = word.split("")
        letters_available = $scrambled_word.split("")

        # Check if word is correct using gem
        if word.correct? && compare_word_arrays(word_to_array, letters_available) && word != ""
            puts "\n#{(" "+ word +" ").upcase.black.on_light_green} is valid.\n\n"
            break
        else
            message = "\n#{(" "+ word +" ").upcase.black.on_red} is invalid. Try another word.\n\n"
        end
    end

end
```

CHECK LETTERS



```ruby
# Check to see if the user used only the available letters
def compare_word_arrays(player_word, letter_pool)

    valid = true

    player_word.each {|c|
        if letter_pool.include?(c)
            letter_pool.delete(c)
        else
            valid = false
        end
    }

    return valid
end
```

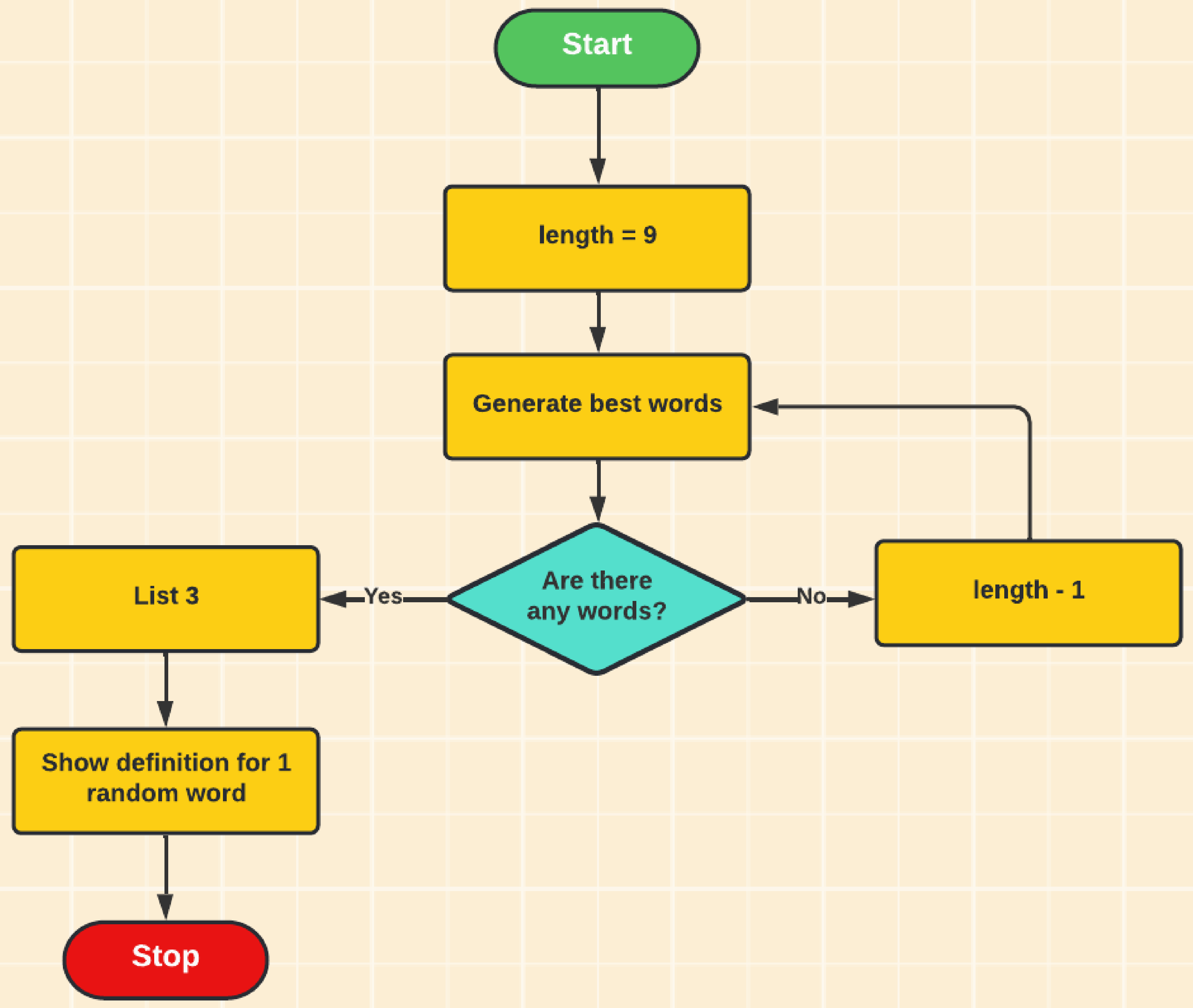
Feature 3.

BEST PLAYABLE WORD

- The best possible words to play are listed
- Only display 3 or fewer words
- The description of 1 word will be given
- Score the player on their word

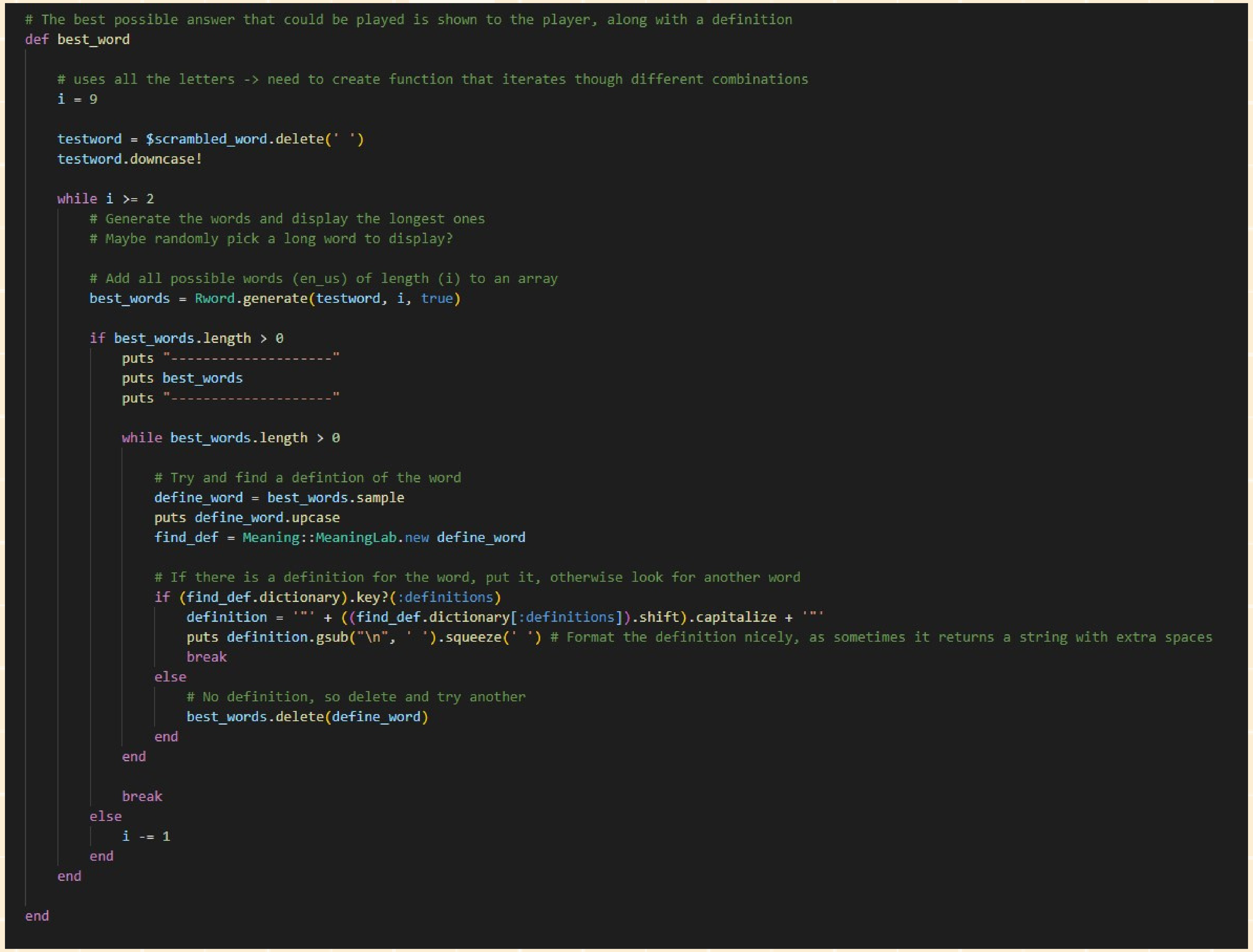
ERROR HANDLING

- Network errors (gems)
- No definition found for best word



Feature 3.

FIND THE BEST WORD



```ruby
# The best possible answer that could be played is shown to the player, along with a definition
def best_word

    # uses all the letters -> need to create function that iterates though different combinations
    i = 9

    testword = $scrambled_word.delete(' ')
    testword.downcase!

    while i >= 2
        # Generate the words and display the longest ones
        # Maybe randomly pick a long word to display?

        # Add all possible words (en_us) of length (i) to an array
        best_words = Rword.generate(testword, i, true)

        if best_words.length > 0
            puts "--------------------"
            puts best_words
            puts "--------------------"

            while best_words.length > 0

                # Try and find a defintion of the word
                define_word = best_words.sample
                puts define_word.upcase
                find_def = Meaning::MeaningLab.new define_word

                # If there is a definition for the word, put it, otherwise look for another word
                if (find_def.dictionary).key?(:definitions)
                    definition = '"' + ((find_def.dictionary[:definitions]).shift).capitalize + '"'
                    puts definition.gsub("\n", ' ').squeeze(' ') # Format the definition nicely, as sometimes it returns a string with extra spaces
                    break
                else
                    # No definition, so delete and try another
                    best_words.delete(define_word)
                end
            end

            break
        else
            i -= 1
        end
    end

end
```

Review.

DEVELOPMENT

- Trello board to layout build process
- Building small

CHALLENGES

- Finding Gems
- Gem incompatibility/errors
- Inappropriate language
- Lack of time

FAVOURITES

- Using gems
- Colorize
- Recreating something

Sanjeev.

TERMINAL APP WALKTHROUGH