# James McGregor

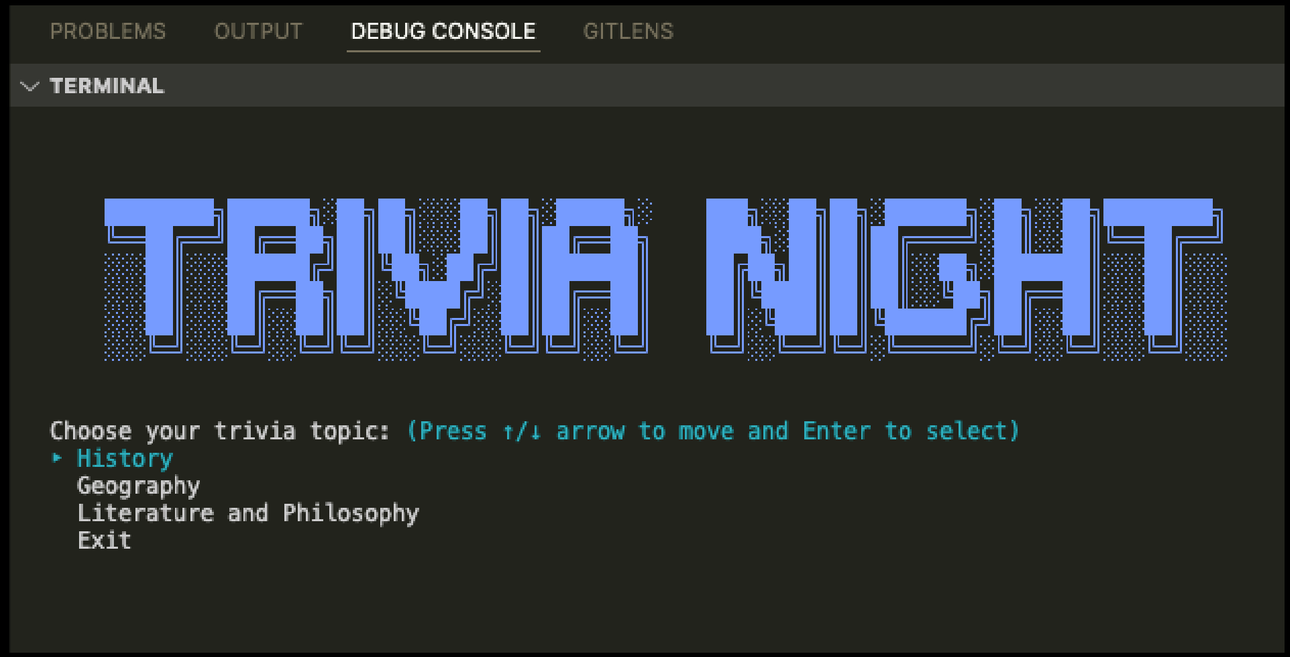
TERMINAL APP

# About the App

## Trivia Game

- Chose something i'm interested in

- Targeting bookworms, nerds, and people generally interested in learning about the world.



# Algorithmic Process + Flow Chart

- Begins with asking the user to stay or leave

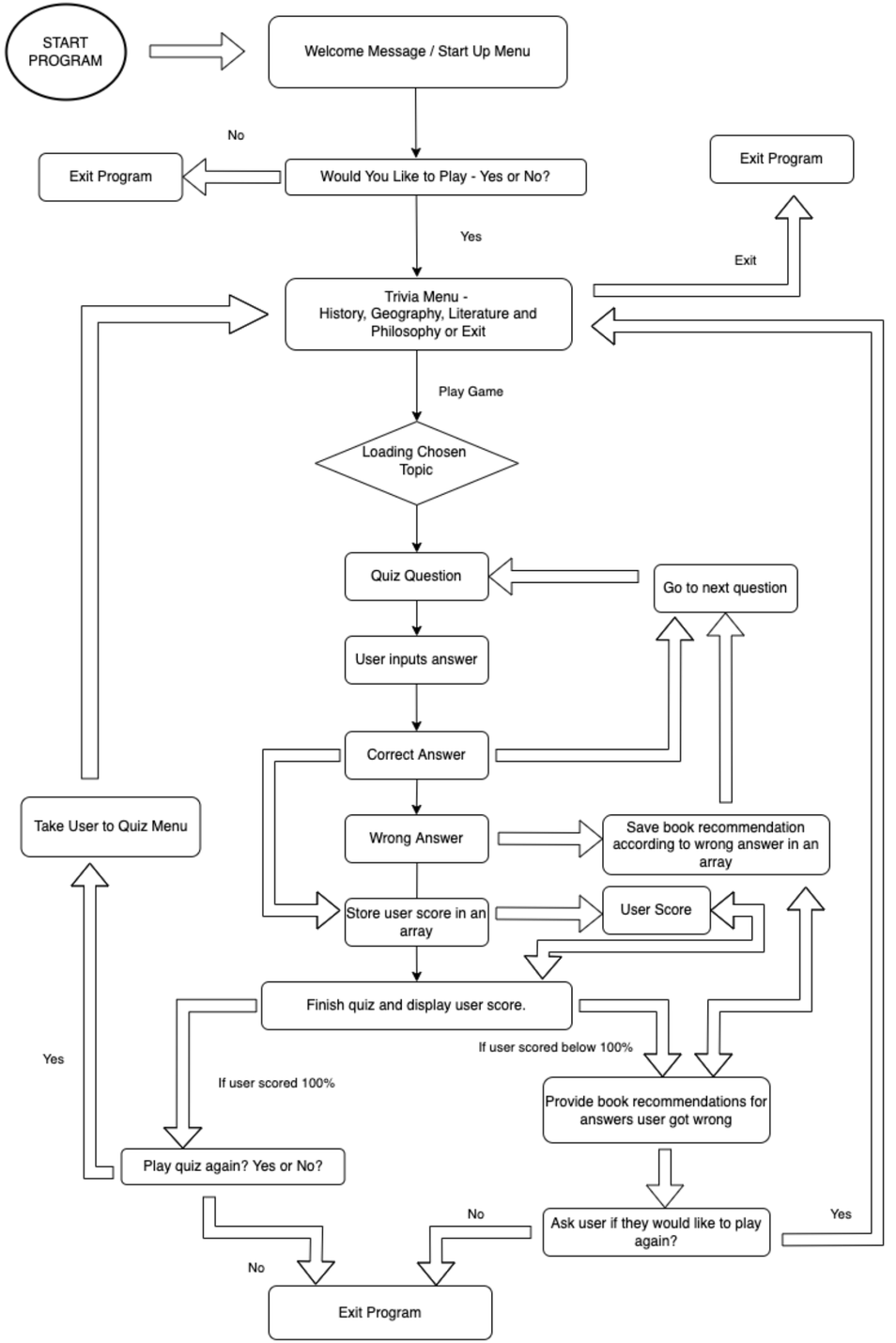
- If user opts to stay, they are provided with a menu of quiz topics

- User will go through quiz - Correct answer, score stored into array; Incorrect answer, book recommendation stored into array.

- When finished quiz, the score will be displayed, with users scoring 100% asked if they want to play again, or exit application.

- Users scoring below 100% asked to view book recommendations.

- User can opt to play again or exit, or view book recommendations.



# Main Program Features

**Examples of screenshots of code and working application can be seen in following slides**

**Feature 1 - Error Handling with TTY Prompt**

TTY Prompt used for error handling on all areas of the application with ask for user input, including - yes or no questions, stay or leave questions, and multiple choice questions

**Feature 2 - Variables**

Variables have been used for all TTY Prompts, TTY Progress Bars, TTY Spinners, Headings/Titles, user score and book recommendation arrays, and topic choices.

**Feature 3 - If Statement and Case Statement**

Case statement used for main menu to choose quiz topics, Multiple If Statements used throughout the program - stay or leave questions, quiz answers (correct and wrong answers), book recommendations to decide whether user needs to receive recommendations or not.

# Overall Structure and Important Logic

**Next few slides will give an overview (in order) of important part of program logic which are central to running the program, and allow everything to work together seamlessly.**

- Welcome page
- Stay or leave option (TTY Prompt)
- Choose quiz topic menu (TTY Prompt, Case Statement)
- Quiz with multiple choice answers (TTY Prompt, multiple If Statements)
- Book recommendations (if program determines user needs recommendations, program uses multiple If statements)
- Return user to quiz menu or exit (if statement, !exit break of code)

# WELCOME PAGE

- ASCII Text Art used to improve the aesthetics of the application from the moment the application is opened.

-- After every system'clear' the title is immediately placed on the screen again. This stops the title from disappearing when new puts statements load on the terminal.

- Basic logic of the application's methods which run after one another (below)

```
argument
startupmenu
stayorleave
choosetopics
```

```ruby
def title
title = "\n\n
    
\n\n".colorize(:light_blue)
end

require "tty-prompt"
require_relative "./main.rb"

def startupmenu
    puts title
    sleep(2)
    puts "\n\nGood evening!!!\n\n".red
    sleep(4)
    system 'clear'
    puts title
    puts "\n\nWelcome to Trivia Night.....\n\n".blue
    sleep(4)
    system 'clear'
    puts title
                        puts "WARNING:".red
    puts "\n\nTHIS APPLICATION IS ONLY FOR PEOPLE WHO ENJOY
PLAYING TRIVIA, OR FOR PEOPLE WHO ENJOY TESTING
THEIR KNOWLEDGE ON INTERESTING AND CHALLENGING TOPICS."
    sleep(10)
    system 'clear'
    puts title
    puts "\n\n"
puts "People who are not interested in playing trivia games may not enjoy this game, lol. "
puts "\nSo if you are one of those people, you are free to leave now ^_^ "
    sleep(6)
    puts "\n\n"
    system 'clear'
end
```

# STAY OR LEAVE

- Use of 'If Statement' inside of a method to ask whether user wants to PLAY or EXIT app.

- TOP RIGHT - Selects EXIT and the 'If Statement' is finished, and the app is terminated.

- BOTTOM - User selects PLAY and the 'If Statement' is finished. Application moves onto execute the "choosetopics" method.

- BOTTOM - Ruby Gem TTY Progress Bar used.

```
Would you like to play or leave? Select below:  Exit

You have chosen to exit the application. Goodbye for now ^_^.
```

```ruby
def stayorleave
    prompt = TTY::Prompt.new
    answer = prompt.select("Would you like to play or leave? Select below: ") do |menu|
        menu.choice 'Play'
        menu.choice 'Exit'
    end

    if answer == 'Play'
        sleep (2)
        puts "\n\nYou want to play? That's great!"
        sleep(3)
        system 'clear'

    else
        sleep(3)
        puts "\n\nYou have chosen to exit the application. Goodbye for now ^_^."
        sleep(5)
        system 'clear'
        exit!
    end

end
```

```
TRIVIA NIGHT

Please wait while we load the trivia menu for you....
Loading [====================      ]
```

```ruby
def choosetopics
    puts title
    puts "\n\nPlease wait while we load the trivia menu for you...."
    puts "\n"
    bar = TTY::ProgressBar.new("Loading [:bar]", total: 30)
    30.times do
    sleep(0.1)
    bar.advance(1)
    end
```

# TRIVIA MENU

- The trivia quiz menu is loaded, and the user is given four choices, provided by the Ruby Gem, TTY Prompt.

- The Case Statement is the code which runs the trivia menu, with the help of the TTY Prompt.

-- After the quiz is finished and user asked to 'stay or leave', if user selects "STAY" the user is thrown out of the cast statement and directly back into the 'choose topics' method (bottom right). This powers the code to bring the user back to the Quiz Menu.

- @availabletopics in 'topics' method used as an array to hold and retrieve the menu choices.

```ruby
def topics
    @availabletopics = ['History', 'Geography', 'Literature and Philosophy', 'Exit']
end

def choosetopics
    puts title
    puts "\n\nPlease wait while we load the trivia menu for you...."
    puts "\n"
    bar = TTY::ProgressBar.new("Loading [:bar]", total: 30)
    30.times do
    sleep(0.1)
    bar.advance(1)
    end

    sleep (4)
    system'clear'
    puts title
    prompt = TTY::Prompt.new
    topicchoice = prompt.select("Choose your trivia topic:", topics, active_color: :cyan, help_color:
:cyan)
```

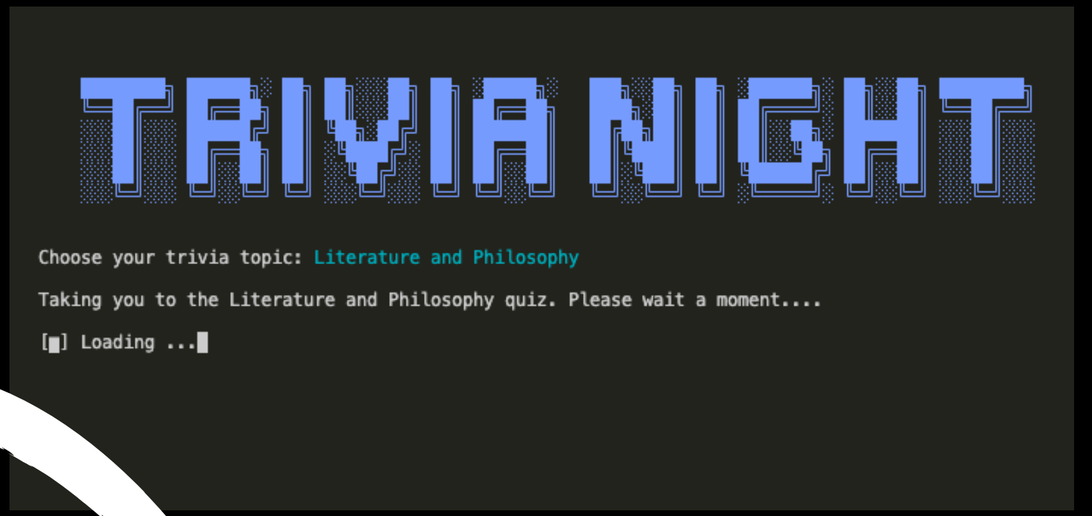

```ruby
case topicchoice

    when 'History'
        loadingquiz "History"
        historyquiz
        stayorleave

    when 'Geography'
        loadingquiz "Geography"
        geographyquiz
        stayorleave

    when 'Literature and Philosophy'
        loadingquiz "Literature and Philosophy"
        litandphilosophyquiz
        stayorleave

    when 'Exit'
        sleep(2)
        puts "\n\nYou selected to exit the application. We hope you return in the future. Goodbye ^_^"
        sleep(5)
        system 'clear'
        exit!
    end

    sleep(1)
    system 'clear'
    choosetopics
end
```

# LOADING SELECTED QUIZ

- Ruby Gem TTY Spinner used to produce a loading timer. Give user the impression the application is actually taking time to load program.

- Used basic method and variable to allow for "DRY" code. If this method was not used, all quiz topics would need all lines of code int his method to be put under each quiz selection.

- This method reduced number of lines of code by 8x3 lines of code.




```ruby
when 'Geography'
    loadingquiz "Geography"
    geographyquiz
    stayorleave

when 'Literature and Philosophy'
    loadingquiz "Literature and Philosophy"
    litandphilosophyquiz
    stayorleave
```

```ruby
def loadingquiz (topic)
    sleep(3)
    puts "\nTaking you to the #{topic} quiz. Please wait a moment....\n"
    puts "\n"
    spinner = TTY::Spinner.new("[:spinner] Loading ...", format: :pulse_2)
    spinner.auto_spin
    sleep(5)
    spinner.stop
    sleep(2)
    system 'clear'
end
```

# PLAYING THE QUIZ

- Ruby Gem TTY Prompt to power the questions and answers for each quiz.

- 'totalscore' important array used to keep track of user's score
- 'histbook' important array used to keep track of the book recommendations.

- Main feature of playing the quiz is the 'If statement' which powers the code when the user is interacting with the quiz.

- Two methods used within the if statement on ALL quiz topics -
  1. def wrong_answer
  2. def correct_answer

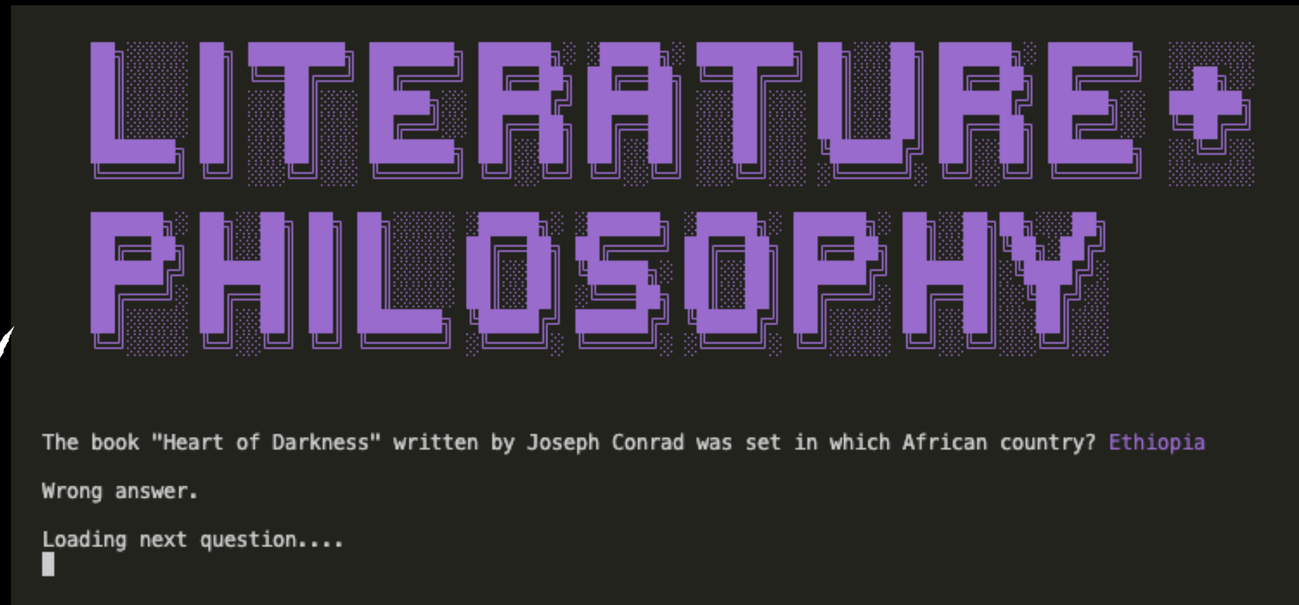
These two methods used to increase the 'DRY'ness of the code.

- 'If statemen't also used to store points in the 'totalscore' array, and the book recommendations in the 'histbook' array.

```ruby
def historyquiz
    totalscore = []
    histbook = []
    $prompt = TTY::Prompt.new
    historytitle
    question1 = 'In what year was the European Union formed?'
    puts "\n\n"
    choices = %w(1917 1993 1945 1905)
    answer = $prompt.select(question1, choices, active_color: :yellow, help_color: :yellow)

    if answer == choices[1]
        sleep(3)
        totalscore << 1
        correct_answer

    else
        histbook << 'You may want to read "The European Union (Politics and Policies)" by Jonathan Olsen to learn more
        about the European Union.'
        wrong_answer
    end
```

```ruby
def wrong_answer
    sleep(3)
    puts "\nWrong answer."
    sleep(3)
    puts "\nLoading next question...."
    sleep(3)
end
```

```ruby
def correct_answer
    puts "\nCorrect answer."
    sleep(3)
    puts "\nLoading next question...."
    sleep(3)
end
```



# SHOWING THE USER'S TOTAL SCORE

```
finishhistory
historytitle
puts "Your total score is #{totalscore.sum} out of 7"
sleep(5)
system 'clear'
```

- When user ends the quiz, the logic of the applpication begins to calculate the score of the user's answers. (Refer to bottom image to see the logic once the last question is answered).

- The total score is produced (top right) by calculating the sum of the 'totalscore' array which was storing the user's score after every correct answer (refer to previous slide).

- Once the user's score has been given to the user, the logic of the program then moves on to the next part of the code, where the program will decide if the user needs to receive book recommendations or not.

- The next slide will explain this next, and final part of the program.

```
def finishhistory
    system 'clear'
    historytitle
    sleep(2)
    puts "Congratulations on finishing the quiz!"
    sleep (3)
    system 'clear'
    historytitle
    puts "Please wait while we calculate your results...."
    puts "\n"
    bar = TTY::ProgressBar.new("Loading [:bar]", total: 30)
    30.times do
    sleep(0.05)
    bar.advance(1)
    end

    sleep(3)
    system 'clear'
end
```

```
else
    sleep(3)
    histbook << 'The book "Meditations" by Marcus Aurelius is one of the best books to get sta
    of the famous Roman Emporers to have ever lived.'
    puts "\nWrong answer."
    sleep(3)
end

    finishhistory
    historytitle
    puts "Your total score is #{totalscore.sum} out of 7"
    sleep(5)
    system 'clear'

if histbook.length > 0
    bookrecommend
    $prompt = TTY::Prompt.new
    question = 'Would you like to view our book recommendations?'
    choices = %w(Yes No)
    answer = $prompt.select(question, choices, active_color: :red, help_color: :red)
    puts "\n"

    if answer == 'Yes'
        sleep(3)
        system 'clear'
        booktitle
        sleep(3)
```

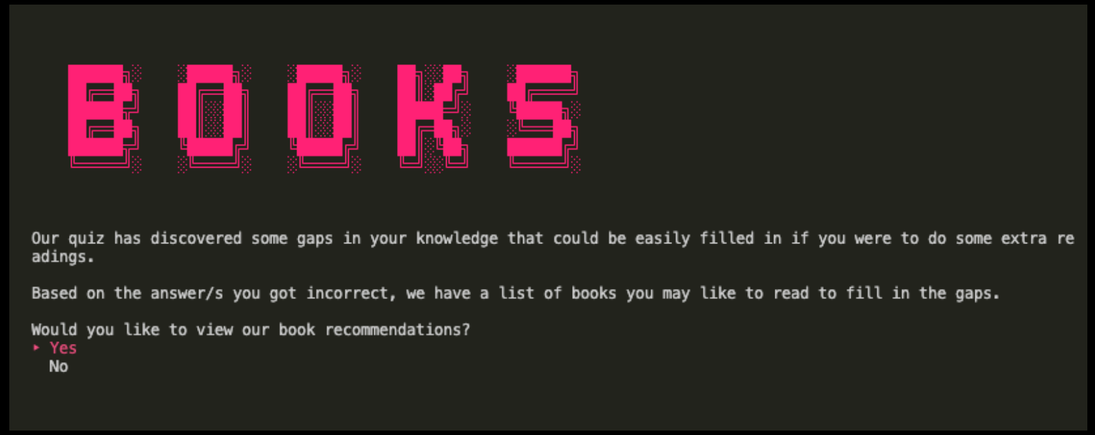
# FINISHING THE QUIZ

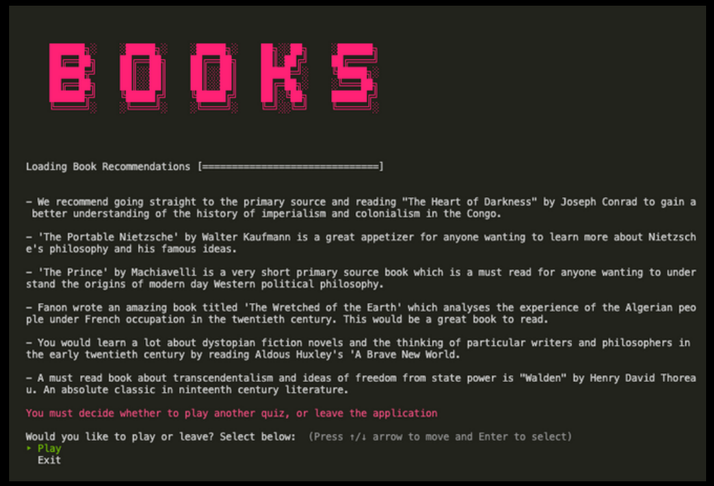
- Main feature of the application with an 'If statement' nested within an 'If statement'.

- The 'If statement' determines whether the 'histbook' array has book recommendations in it. This means that the user has answered at least one question incorrectly.

- The program then asks the user if they would like to view the book recommendations, powered by the Ruby Gem TTY Prompt.

- This is one of the more complex features of the application in terms of code and logic (for someone at my level of experience).




```ruby
if histbook.length > 0
    bookrecommend
    $prompt = TTY::Prompt.new
    question = 'Would you like to view our book recommendations?'
    choices = %w(Yes No)
    answer = $prompt.select(question, choices, active_color: :red, help_color: :red)
    puts "\n"

    if answer == 'Yes'
        sleep(3)
        system 'clear'
        booktitle
        sleep(3)
        puts "\n"
        bar = TTY::ProgressBar.new("Loading Book Recommendations [:bar]", total: 30)
        30.times do
        sleep(0.05)
        bar.advance(1)
        end
        puts "\n\n"
        sleep(3)
        histbook.each do |item|
        puts "- #{item}"
        puts "\n"
        sleep(3)
        end

    else
        sleep(3)
        puts "Please wait while we take you back to the main menu...."
        sleep(4)
        system 'clear'
    end
```

# PLAY AGAIN OR LEAVE?

- Once user has finished receiving book recommendations, logic of the program will move onto the 'stayorleave' method once again - user will be asked to play agan or leave the application

- The logic of the 'stayorleave' method will allow the user to stay or leave by executing two different actions.

- If the user selects 'yes' to stay, it will jump the user out of the 'case statement', but it is still inside the choose topics method, and inside the choosetopics method, below the case statement the 'choosetopics' method is called AGAIN so it automatically returns the user to the quiz topics menu. (refer to the top image with "choosetopics" outside of the case statement at the bottom of the code snippet.)

- If the user decides to leave, the '!exit' function will be executed, and the application will be closed.

```
        when 'History'
            loadingquiz "History"
            historyquiz
            stayorleave

        when 'Geography'
            loadingquiz "Geography"
            geographyquiz
            stayorleave

        when 'Literature and Philosophy'
            loadingquiz "Literature and Philosophy"
            litandphilosophyquiz
            stayorleave

        when 'Exit'
            sleep(2)
            puts "\n\nYou selected to exit the appli
            sleep(5)
            system 'clear'
            exit!
    end

    sleep(1)
    system 'clear'
    choosetopics
```

```
def stayorleave
    prompt = TTY::Prompt.new
    answer = prompt.select("Would you like to play or leave? Select below: ") do |menu|
        menu.choice 'Play'
        menu.choice 'Exit'
    end

    if answer == 'Play'
        sleep (2)
        puts "\n\nYou want to play? That's great!"
        sleep(3)
        system 'clear'

    else
        sleep(3)
        puts "\n\nYou have chosen to exit the application. Goodbye for now ^_^."
        sleep(5)
        system 'clear'
        exit!
    end

end
```

# STORING APPLICATION HEADINGS

- The headings stored in a separate .rb file to increase the 'DRY'ness of the code in the application.

- The headings are called on whenever they are needed throughout the application.

```ruby
if answer == 'Yes'
    sleep(3)
    system 'clear'
    booktitle
    sleep(3)
    puts "\n"
    bar = TTY::ProgressBar.new("Loading Book Recommendations [:bar]",
    30.times do
    sleep(0.05)
```

```ruby
def historyquiz
    totalscore = []
    histbook = []
    $prompt = TTY::Prompt.new
    historytitle
    question1 = 'In what year was the European Union formed?'
    puts "\n\n"
```

```ruby
def booktitle
    puts "\n\n
    \n\n".colorize(:light_red)
end

def historytitle
    puts "\n\n
    \n\n".colorize(:yellow)
end

def geographytitle
    puts "\n\n
    \n\n".colorize(:green)
end
```

# Thank you for viewing my presentation

JAMES
MCGREGOR