JARED GOLDSTEIN

CASINO

Ruby
Terminal
Application

Purpose

Are you : Bored?

Bad with Money?

Need to quit Gambling?

Stuck in Quarantine?

If answer == 'yes' || 'y'

I have made the perfect app for you!

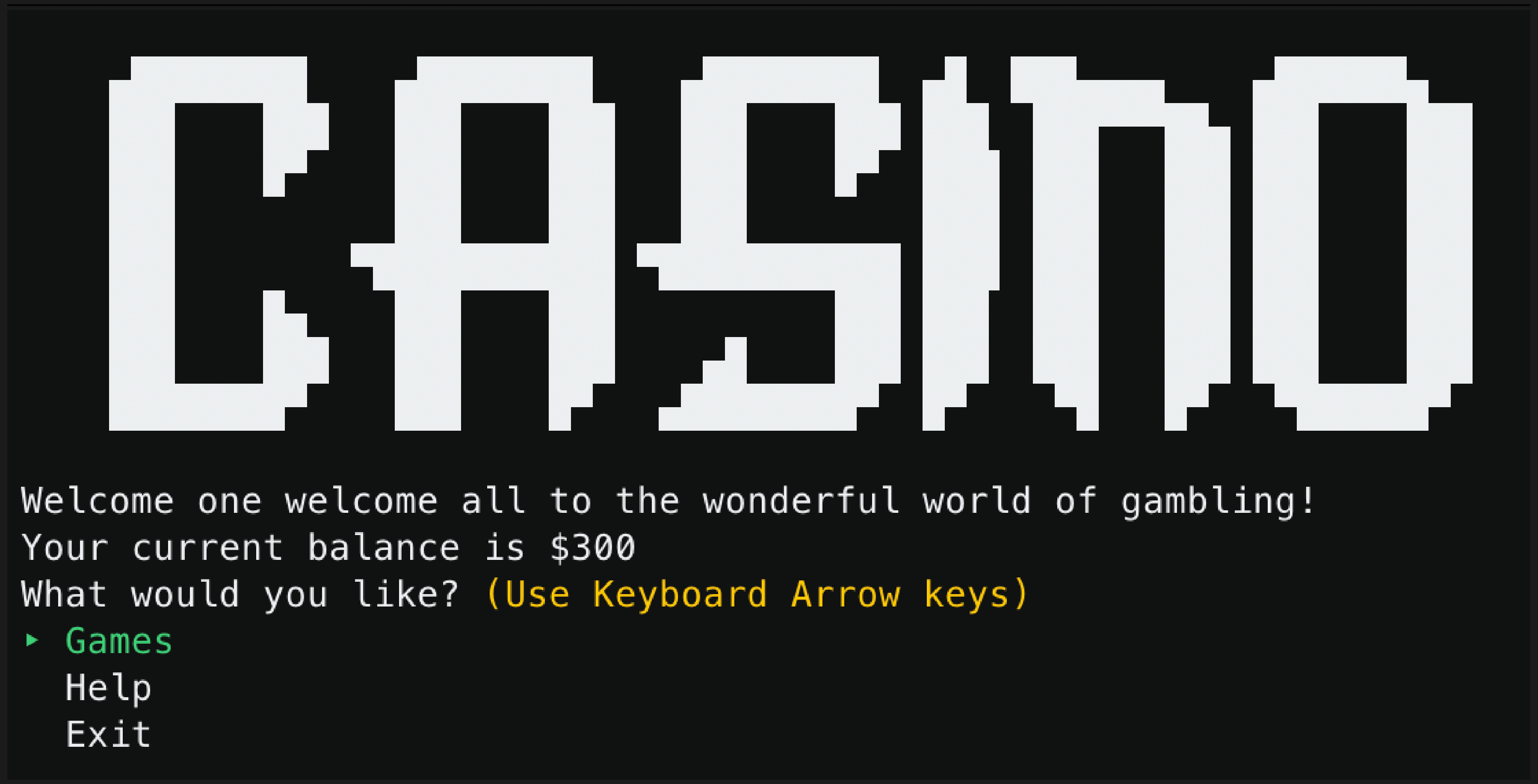
What is it?

Casino is a collection of games you may find in a Casino or other gambling estabilishments. It Includes:

1. Blackjack
2. Crash
3. Horse Racing

How it works

Casino starts by introducing the user to their wallet which is valued at $100. They are then given options like if they need help or would like to play one of the three games.



```
Welcome one welcome all to the wonderful world of gambling!
Your current balance is $300
What would you like? (Use Keyboard Arrow keys)
‣ Games
  Help
  Exit
```

Ruby
Terminal
Application

How it works

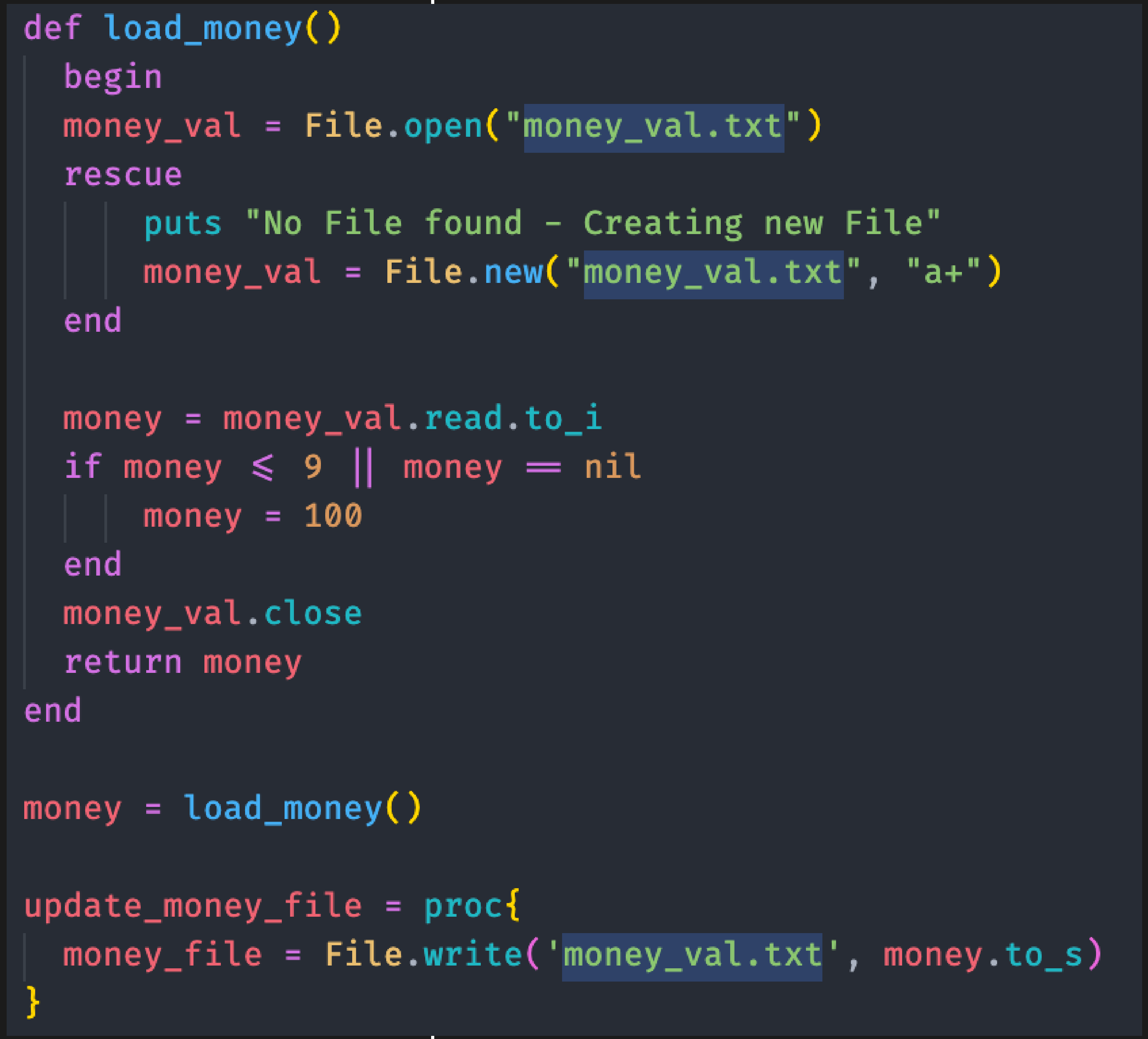
When in the user chooses to play one of the games they are first introduced to learn more about the games rules. Alternatively they can continue on and play the game.

Each game has different logic so for this I will go through some key features and explain Blackjacks logic. Afterwhich I will give a demonstration.

About Casino

Simple and Replayable

Using File reading and writing my application saves your money value allowing infinite replayability



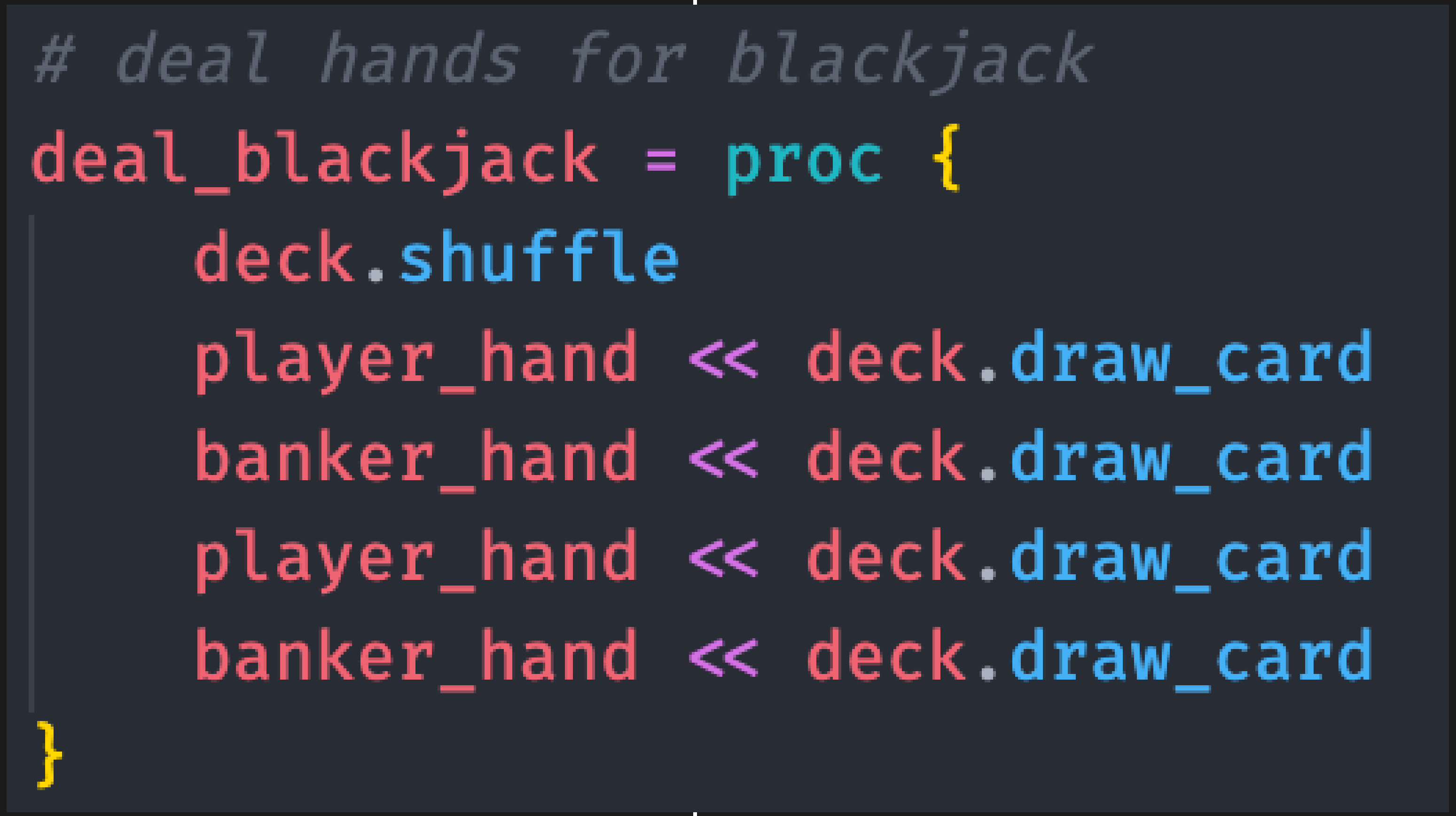
```ruby
def load_money()
  begin
  money_val = File.open("money_val.txt")
  rescue
      puts "No File found - Creating new File"
      money_val = File.new("money_val.txt", "a+")
  end

  money = money_val.read.to_i
  if money <= 9 || money == nil
      money = 100
  end
  money_val.close
  return money
end

money = load_money()

update_money_file = proc{
  money_file = File.write('money_val.txt', money.to_s)
}
```

MADE FOR ADDICTION

Ruby
Terminal
Application

Fair Blackjack

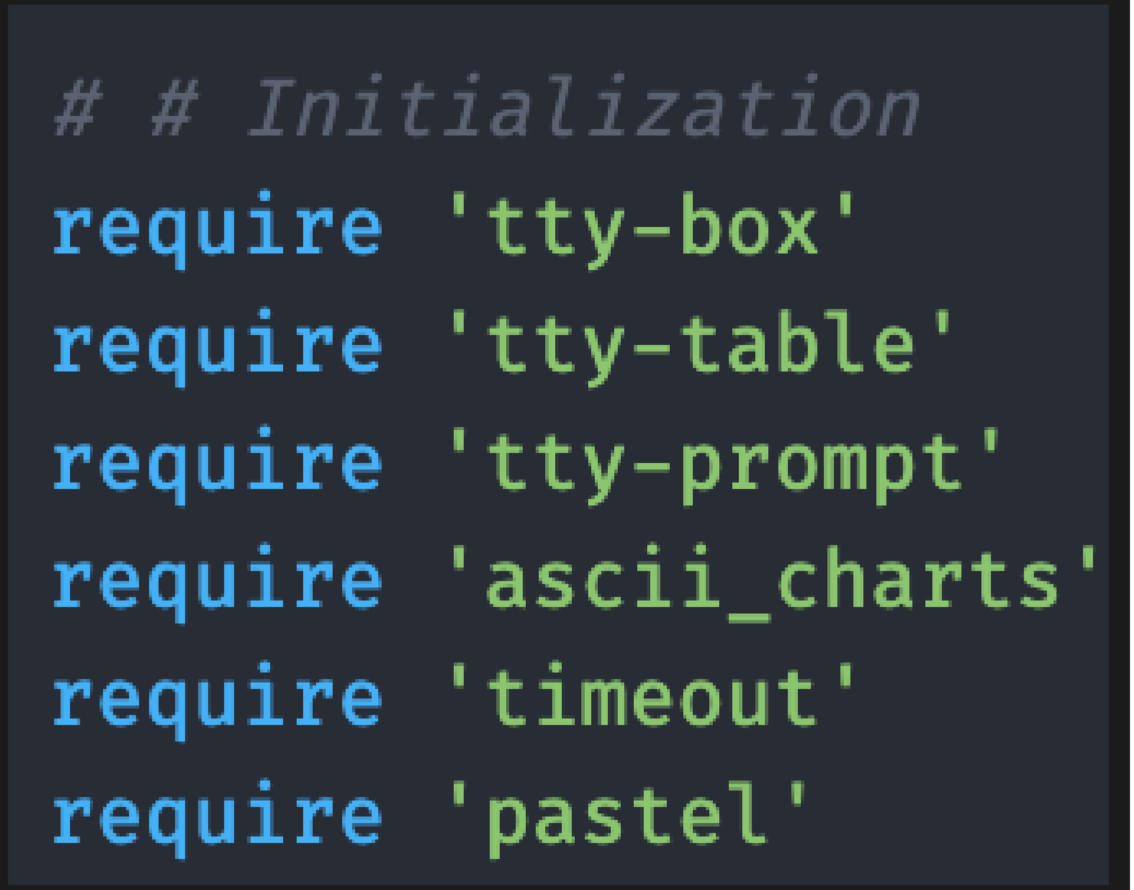
Using similar structure I have created blackjack the same way you would be able to play in a Casino. This means you're dealt cards the correct way and you are given similar opportunities as if you were actually there.



```ruby
# deal hands for blackjack
deal_blackjack = proc {
    deck.shuffle
    player_hand << deck.draw_card
    banker_hand << deck.draw_card
    player_hand << deck.draw_card
    banker_hand << deck.draw_card
}
```

Simple User Inputs

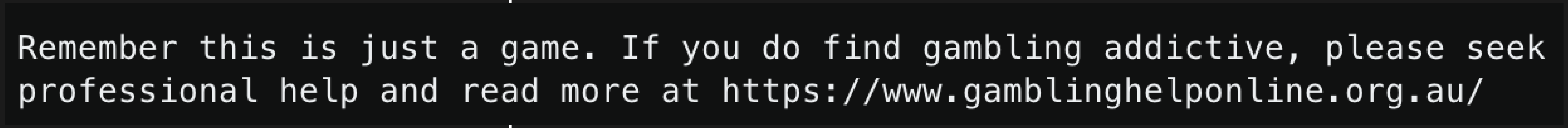
I have taken advantage of many gems to give the user a simple and clear browsing experience.



```ruby
# # Initialization
require 'tty-box'
require 'tty-table'
require 'tty-prompt'
require 'ascii_charts'
require 'timeout'
require 'pastel'
```

Ethical Concerns

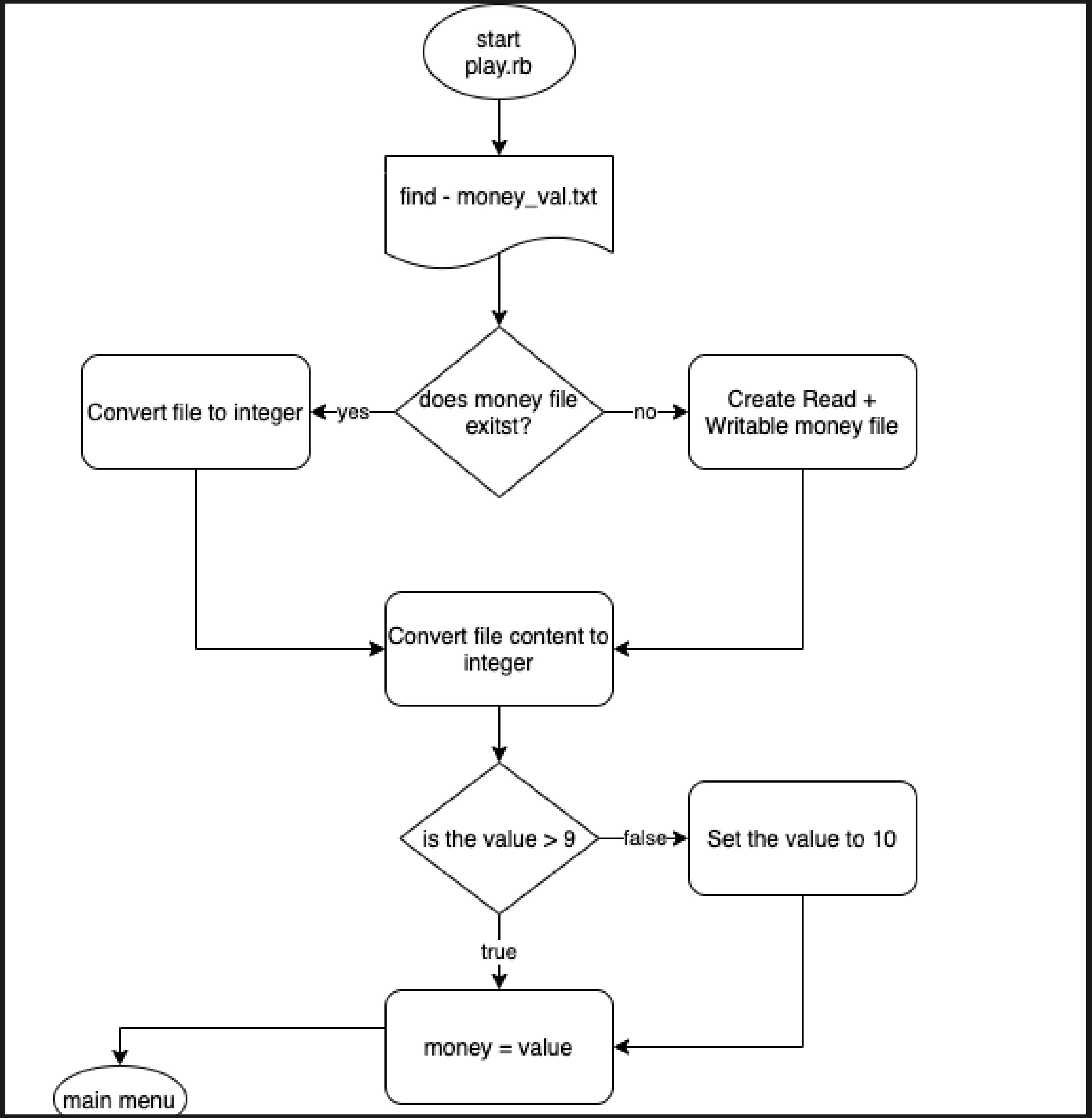
Finally to addres the most important topic relating to a gambling application. My app raises some major ethical issues by introducing or tempting users with the addiction of gambling. With this I have given appropriate gambling help links to guide the user to help if they require it.



```
Remember this is just a game. If you do find gambling addictive, please seek
professional help and read more at https://www.gamblinghelponline.org.au/
```

//09

How it works



start
play.rb

find - money_val.txt

does money file exitst?

yes

Convert file to integer

no

Create Read + Writable money file

Convert file content to integer

is the value > 9

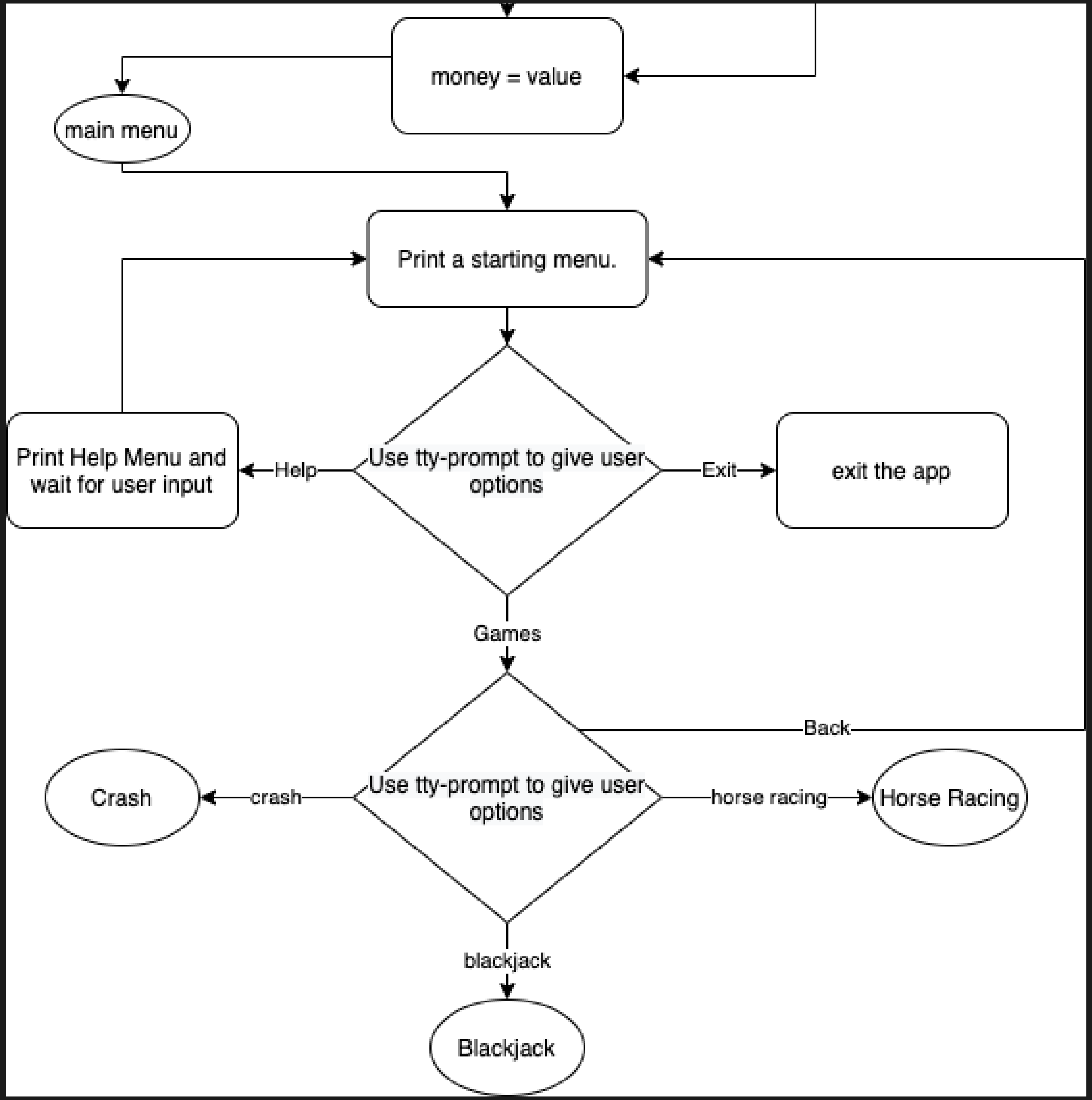
false

Set the value to 10

true

money = value

main menu



money = value

main menu

Print a starting menu.

Use tty-prompt to give user options

Help

Print Help Menu and wait for user input

Exit

exit the app

Games

Use tty-prompt to give user options

Back

crash

Crash

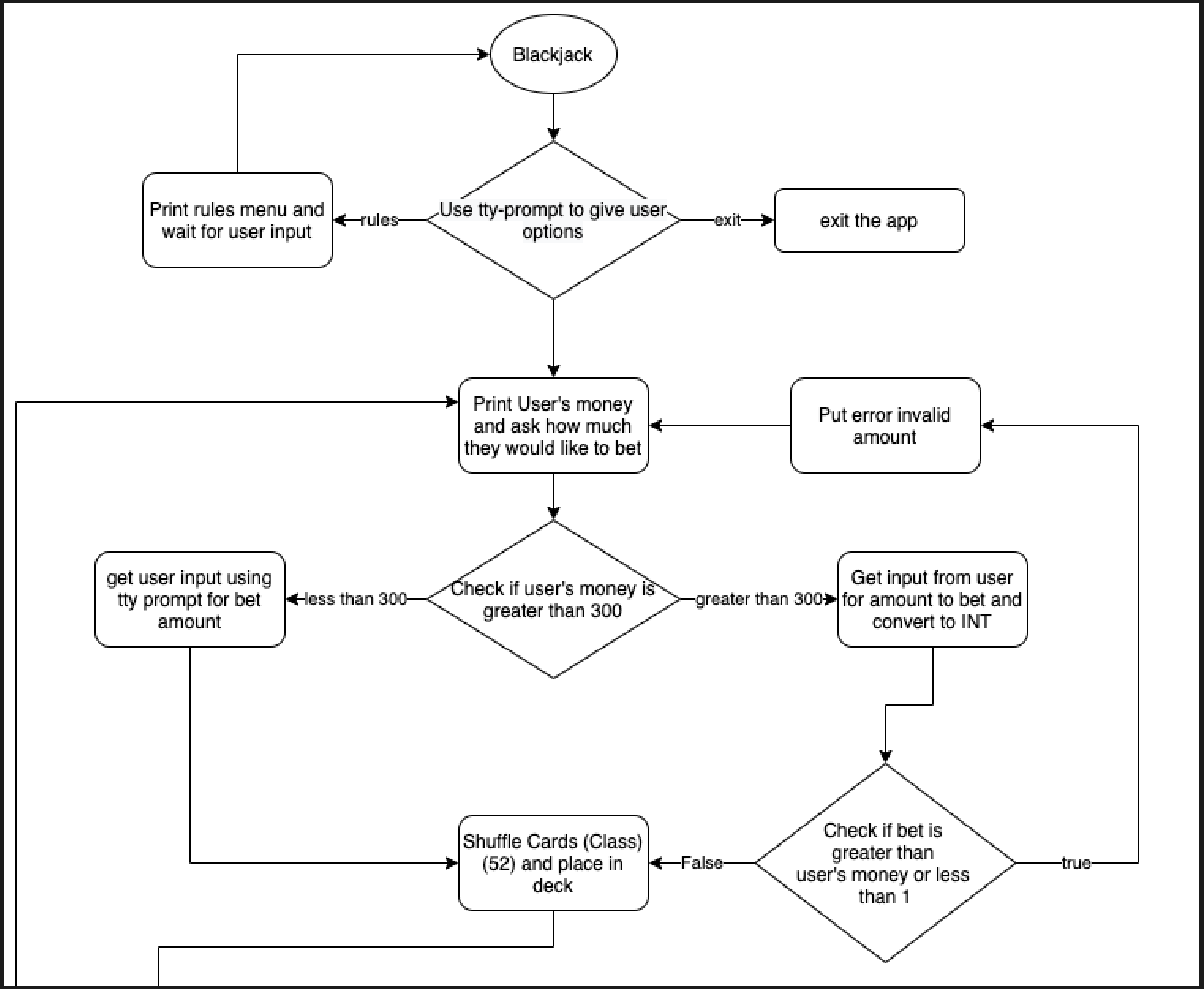
horse racing

Horse Racing

blackjack

Blackjack

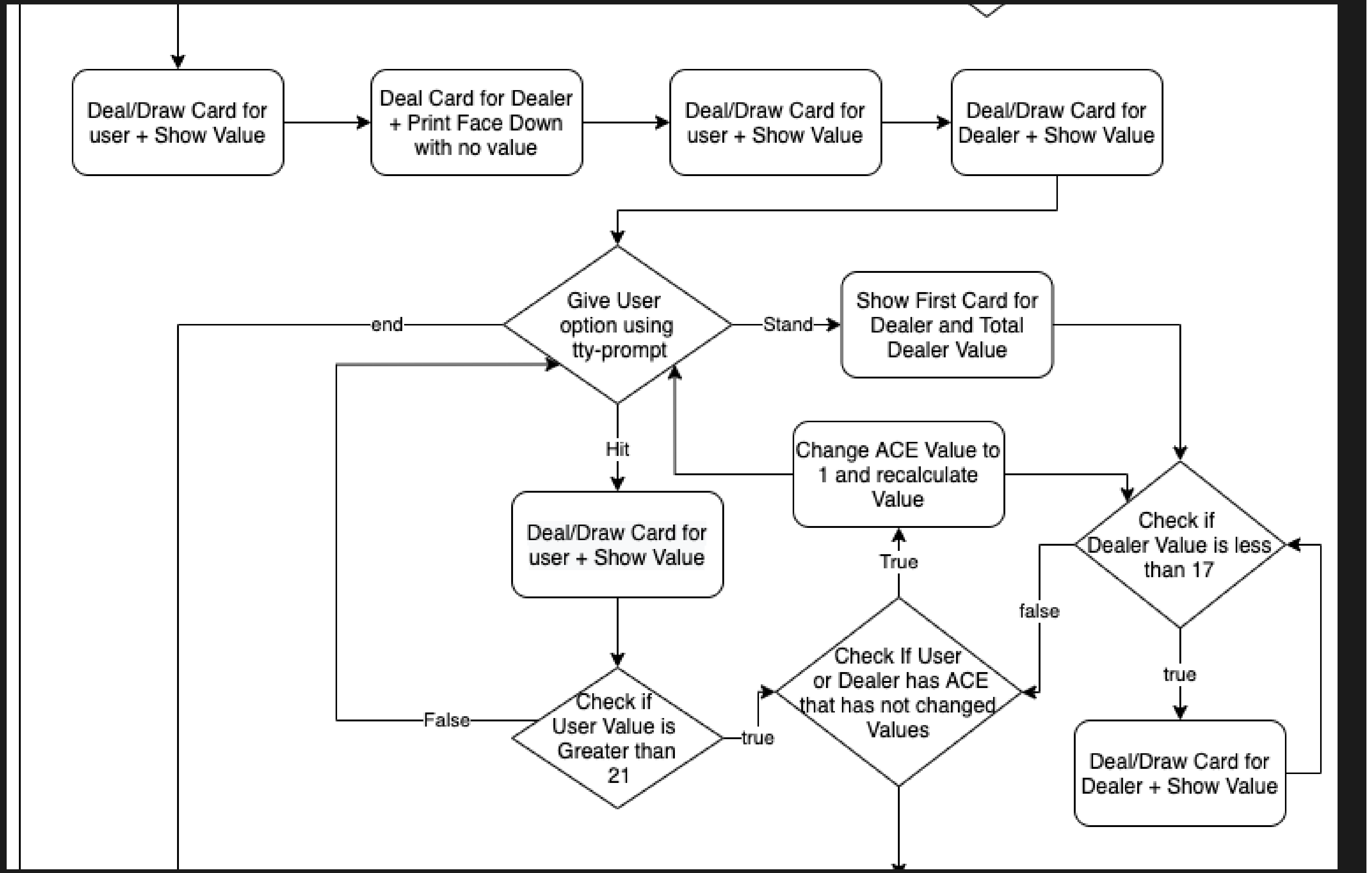
How it works

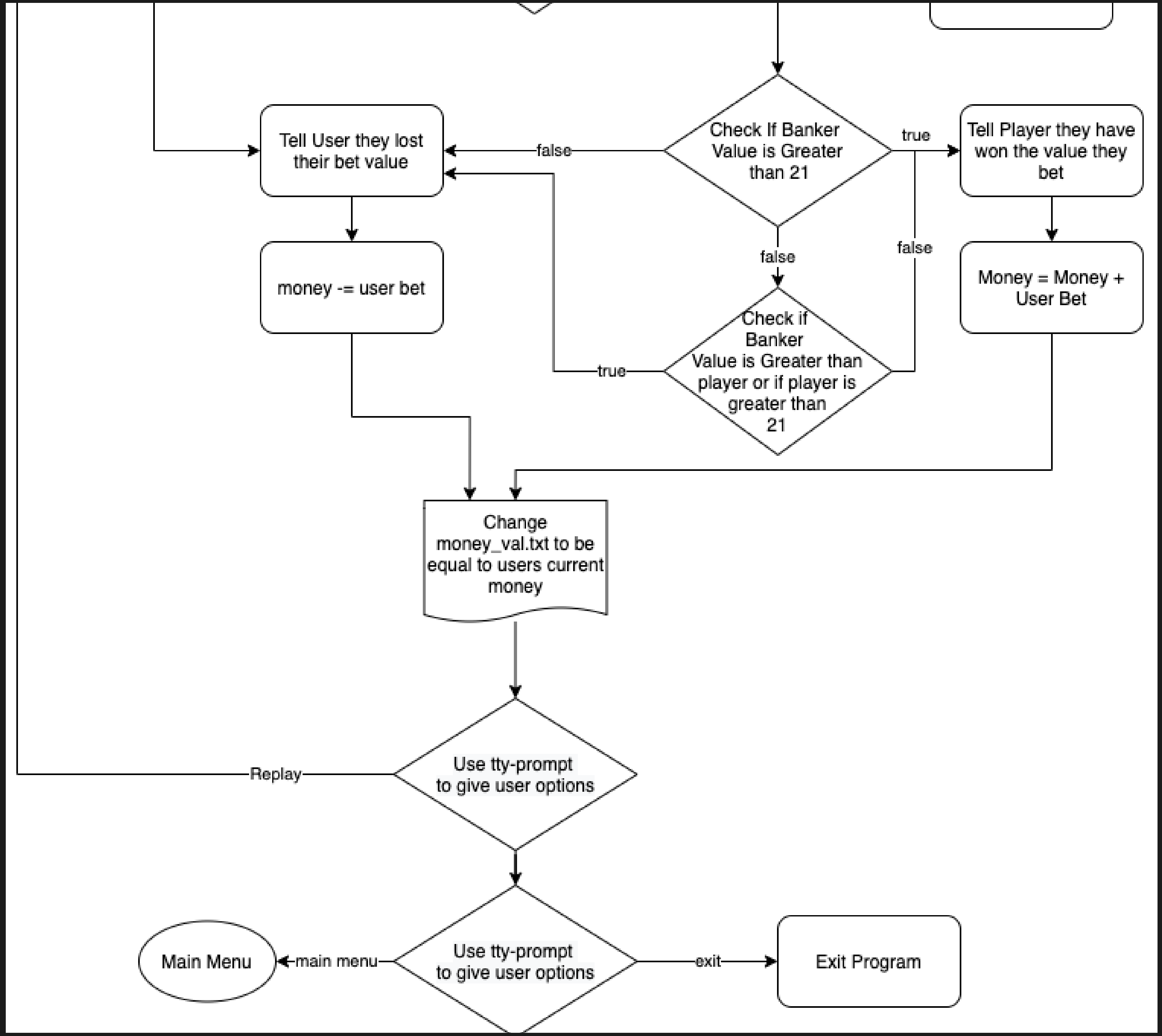


Blackjack
Use tty-prompt to give user options
rules
Print rules menu and wait for user input
exit
exit the app
Print User's money and ask how much they would like to bet
Put error invalid amount
Check if user's money is greater than 300
less than 300
get user input using tty prompt for bet amount
greater than 300
Get input from user for amount to bet and convert to INT
Check if bet is greater than user's money or less than 1
true
False
Shuffle Cards (Class) (52) and place in deck

//11

How it works


Deal/Draw Card for user + Show Value
Deal Card for Dealer + Print Face Down with no value
Deal/Draw Card for user + Show Value
Deal/Draw Card for Dealer + Show Value
Give User option using tty-prompt
end
Stand
Show First Card for Dealer and Total Dealer Value
Hit
Change ACE Value to 1 and recalculate Value
Deal/Draw Card for user + Show Value
True
Check if Dealer Value is less than 17
false
true
Check If User or Dealer has ACE that has not changed Values
Check if User Value is Greater than 21
False
true
Deal/Draw Card for Dealer + Show Value


Tell User they lost their bet value
false
Check If Banker Value is Greater than 21
true
Tell Player they have won the value they bet
money -= user bet
false
false
Money = Money + User Bet
true
Check if Banker Value is Greater than player or if player is greater than 21
Change money_val.txt to be equal to users current money
Replay
Use tty-prompt to give user options
Main Menu
main menu
Use tty-prompt to give user options
exit
Exit Program

Things I Liked

- The Ruby syntax was very fun to work with.
- Having multiple goals and being able to make them all
- Creating something that I enjoyed playing throughout the process
- Ruby gems were a lot of fun to look into and find what worked.
- Really cementing what I learnt in a physical application

Struggles

- Test Driven development was quite a hard method for me to use
- Setting a large/small enough scope made it difficult
- Keeping to a strict creation timeframe and structure

Now time for a Demonstation!