CARD GAME

By John Okinedo

Cards game Project by John Okinedo:

Description of the system:

Before the game starts, the user must enter a username and password in order to be allowed access to start the game the login details will be stored in a file that is stored on the system. This game that I’m creating involves 3 different colour cards, which each have a number from 1 – 10 on them. Player 1 takes a card from the cards and player 2 does the same. If they draw the same colour then the program will look which has the highest number on the card. The deck is also shuffled before the 1st round starts. The player who wins the most rounds wins the game.

# Flowchart of early game design

Diagram

Description automatically generated

# Psuedocode

Importing random, mysql.connector, tkinter and PIL,CSV

#FRONT END FUNCTIONS

def login\_screen():

login\_window = Tk()

login\_window.geometry("253x132")

login\_window.configure(background="purple")

login\_window.title("Login")

name\_label = Label(login\_window,text = "Name: ",bg="#9489f7")

name\_label.grid(row = 0, column = 0)

password\_label = Label(login\_window,text = "Password: ",bg="#9489f7")

password\_label.grid(row = 1, column = 0)

user\_login = Entry(login\_window)

user\_login.grid(row = 0, column = 1)

pass\_login = Entry(login\_window,show = "\*")

pass\_login.grid(row = 1, column = 1)

enter\_button = Button(login\_window,text="login", command= authentication(user\_login,pass\_login,login\_window),bg="#da001c")

enter\_button.grid(size=3)

login\_window.mainloop()

return user\_login, pass\_login,login\_window

def decide\_winner():

update\_score()

if p1\_card\_total > p2\_card\_total:

OUTPUT "player 1 wins”

winner = p1\_card\_total

elif p1\_card\_total < p2\_card\_total:

OUTPUT “Player 2 wins”

winner = p2\_card\_total

else:

OUTPUT “tie”

winner = p1\_card\_total

score\_board deleted

window\_to\_ask\_for\_name(winner)

def rules\_section()

output rules.txt

def main\_menu():

main\_menu\_window = Tk()

main\_menu\_window.configure(background="light blue")

main\_menu\_window.title("Main Menu")

main\_menu\_window.geometry("300x200")

welcome\_tag = Label(main\_menu\_window,text = "Welcome to the card game")

play\_button = Button(main\_menu\_window,text = "Draw Cards", command= draw\_cards)

leaderboard\_button = Button(main\_menu\_window,text="Leaderboard", command=leaderboard)

rules\_button = Button(main\_menu\_window,text= "How to play", command=rules\_section)

quit\_button = Button(main\_menu\_window,text="Quit", command=quit)

main\_menu\_window.update()

def change\_colour():

colour = random colours

play\_button = random colours

change\_colour()

def update\_score():

p1\_score\_label.config(text=p1\_card\_total)

p2\_score\_label.config(text=p2\_card\_total)

score\_board.update()

def window\_to\_ask\_for\_name(winner):

name\_window = Tk()

name\_window.geometry("600x100")

name\_window.configure(background="white")

ask\_name = Label(name\_window,text="Enter your name to be added to the leaderboard (max 5 letters)”)

enter\_name = Entry(name\_window)

def add\_to\_leaderboard():

name = enter\_name.get()

length = length of name

if length <= 5:

update\_leaderboard(winner,name)

delete name\_window

elif length > 5 or length == 0:

OUTPUT("Error","Name is too long, please try again")

name\_button = Button(name\_window,text="Add to leaderboard",command=add\_to\_leaderboard)

name\_window.mainloop()

def update\_leaderboard(winner,name):

file = open("Leaderboard.csv","a")

INPUT name and score

file.close()

def leaderboard():

x = 1

leaderboard\_window = Tk()

leaderboard\_window.geometry = ("150x100")

leaderboard\_window.title = "Leaderboard"

open(“leaderboard.csv”,”r”

listbox = Listbox(leaderboard\_window)

listbox.insert("Pos Name Score")

for names in top 5 in leaderboard

output names

x = x + 1

leaderboard\_window.mainloop()

BACK END

def authentication(user\_login, pass\_login,login\_window):

open sql database

username = user\_login.get()

passcode = pass\_login.get()

sql\_query = "SELECT \* FROM info WHERE username = username AND password = passcode”

if sql\_query = username and password:

OUTPUT ("Login successful Welcome back username!")

main\_menu()

delete login window

else:

OUTPUT ("Login failed Incorrect username or password, please try again.")

def draw\_cards():

try:

global p1\_card, p2\_card

p1\_card = [first card]

delete p1\_card from deck

p2\_card = [second card]

delete p2\_card card from deck

card\_images = [ dictionary of the colours, numbers and it links to the image file of the cards] ]

for card in card\_images:

if card["colour"] == colour\_of\_card and card["number"] == number\_of\_card:

p1\_pic\_card = Image.open(card["img"])

photo = ImageTk.PhotoImage(p1\_pic\_card)

p1\_display\_drawn\_card.config(image=photo)

p1\_display\_drawn\_card.image = photo

break

OUTPUT "Player 1 drew {p1\_card} and player 2 drew {p2\_card}”

compare\_cards(p1\_card, p2\_card)

except if deck is empty:

OUTPUT ("There are no more cards, to play again restart the game")

def compare\_cards(p1\_card, p2\_card):

global p1\_card\_total, p2\_card\_total

if colours are the same:

if player 1 number is higher than player 2’s:

OUTPUT "👑Player 1 wins!!!👑"

p1\_card\_total += 2

p1\_winning\_cards.append(p1\_card)

p1\_winning\_cards.append(p2\_card)

elif player 1’s number is less than player 2’s

OUTPUT "👑Player 2 wins!!!👑”

p2\_card\_total += 2

p2\_winning\_cards.append(p2\_card)

p2\_winning\_cards.append(p1\_card)

else:

if player 1 colour > player 2 colour

OUTPUT "👑Player 1 wins!!!👑"

p1\_card\_total += 2

p1\_winning\_cards.append(p1\_card)

p1\_winning\_cards.append(p2\_card)

else:

OUTPUT "👑Player 2 wins!!!👑”

p2\_card\_total += 2

p2\_winning\_cards.append(p2\_card)

p2\_winning\_cards.append(p1\_card)

next\_round()

update\_score()

def next\_round():

if deck == None:

decide\_winner()

#main program

p1\_card\_total = 0

p2\_card\_total = 0

score\_board = Tk()

score\_board.title("Current Score")

score\_board.geometry("500x400")

score\_board.configure(background="grey")

p1\_title = Label(score\_board,text="player 1 score")

p1\_score\_label= Label(score\_board,text=p1\_card\_total)

p2\_title = Label(score\_board,text="player 2 score”)

p2\_score\_label = Label(score\_board,text=p2\_card\_total,bg="grey")

cards\_drawn = Label(score\_board,text="",bg="grey")

p1\_display\_drawn\_card = Label(bg="grey")

p2\_display\_drawn\_card = Label(bg="grey")

winner\_of\_round = Label(bg="grey")

p1\_display\_drawn\_card.place(x=100,y=200)

p2\_display\_drawn\_card.place(x=350,y=200)

p1\_winning\_cards = []

p2\_winning\_cards = []

colours = ["red", "yellow", "black"]

numbers = range(1, 11)

deck = [(colour, number) for colour in colours AND for number in numbers]

r.shuffle(deck)

login\_screen()

# function and variable listing for final program:

## function list

login\_screen() = this is the login screen where a login window will be displayed, take in the username and password and return them into the main program and into the authentication() function

decide\_winner() = this will update the score board for the final time and display a messagebox saying who the winner was and show how many cards (the score) they got, it will then destroy the score\_board window and assign the winners score to winner it will then open the window\_to\_ask\_for\_name function

rules\_section() = this opens the rules text file

main\_menu() = this opens the main menu window where the user will be able to select to draw their cards, open the leaderboard, open the rules or exit the game

change\_score() = this changes the colour of the “draw cards” button to make it more appealing to the user to press

update\_score() = this updates the score labels of both player 1 and player 2

window\_to\_ask\_for\_name(winner) = this opens a window to allow the winner to enter their name to be added to a leaderboard

add\_to\_leaderboard() = this is a function inside the previous function, this takes the name of the winner and the score of the winner passed previously and passes them into another function called update\_leaderboard(winner,name)

update\_leaderboard(winner,name) = this is where the name of the winner and the score are added to the csv file

leaderboard() = this opens the leaderboard csv file in a separate window, reads the whole csv and outputs the players who are in the top 5 in terms of high score into a listbox

authentication(user\_login, pass\_login, login\_window) = this opens the mysql database to check if the user and password entered by the user are the same as the ones stored in the database, if they are then a messagebox will be presented greeting the user but if its not then a error box will be displayed to let the user to know that those are the incorrect credentials and should try again

draw\_cards() = this uses try and except where in the try the first card will be assigned to player 1 and the second card will be assigned to player 2, these assigned cards will then be deleted from the deck afterwards the program checks in the card\_images dictionary that if the colour of the card and the number of the card are the same as the one drawn then the program will display the image of that respective card. Now for the except if the deck is empty and the player tries to draw cards anymore cards then an error message will display saying that there are no more cards available.

Compare\_cards(p1\_card, p2\_card) = this is a function that will compare each cards and decide who wins the round, this will then open the next\_round() function and the update\_score() function

Next\_round() = this will check if the deck is empty to continue, if it is empty then it will open the decide\_winner() function

## Variable list:

P1\_card\_total & p2\_card\_total = this is a global variable that holds the current score of the players

P1\_card & p2\_card = the cards drawn by the players

Colours = this list holds the colours of the the cards

Numbers = choses a random number from 1 to 11 to be stored in the list

Deck = this is a global variable that holds the cards in a list

Winner = the score of the winner

Colour = changes the colour of the button randomly

Username = the username entered by the user in the login screen

Passcode = the password entered by the user in the login screen

Sql\_query = the sql query used to search for the username and password in the program

Result = the result of the sql query

Db = this is used to connect to the mysql database

Card\_images = a dictionary storing the colour and number of each card possibility, this then allocate them a image that correspond to the card combination

Login\_window = this is the window for the login screen

Password\_label = this is a label to say where the password is on the login screen

Name\_label = this is a label to say where the username is on the login screen

Main\_menu\_window = this is the window for the main menu

Font\_for\_game = this is a variable to decide what font, size and weight each letter is in the game

Welcome\_tag = this is a welcome message at the top of the program

Play\_button = this is a button, that will open the draw\_cards() function when pressed

Leaderboard\_button = this is a button that will open the leaderboard() function when pressed

Rules\_button = this is a button that will open the rules() function when pressed

Quit\_button = this is a button that will close the program when pressed

Length = this is to see the length of the name entered to be added to the leaderboard

File = used to open the leaderboard.csv file, as this was the only file opened in the whole program I felt calling it file was fine

Reader = this is a variable that reads the leaderboard file

Sorted\_data = this sorts the leaderboard file in descending order in order of the players score

Listbox = the listbox used for to display the leaderboard

P1\_winning\_cards & p2\_winning\_cards = this is a global variable that holds the cards won by the players

P1\_display\_drawn\_cards & p2\_display\_drawn\_cards = this is a label in tkinter that is used to display the images of the cards

# Actual Code

import random as r

import mysql.connector

from tkinter import \*

from tkinter import messagebox

from tkinter.font import Font

from PIL import Image,ImageTk

#===========================================================FRONT-END=========================================================================================#

def login\_screen():

    #here im using tkinter to create the login screen

    login\_window = Tk()

    login\_window.geometry("253x132")

    login\_window.configure(background="#9489f7") #used hexadecimal to get a specific type of purple background

    login\_window.title("Login")

    name\_label = Label(login\_window,text = "Name: ",bg="#9489f7")

    name\_label.grid(row = 0, column = 0) #i used rows and columns instead of .pack() as this way it made the login screen look more organised

    password\_label = Label(login\_window,text = "Password: ",bg="#9489f7")

    password\_label.grid(row = 1, column = 0)

    user\_login = Entry(login\_window)

    user\_login.grid(row = 0, column = 1)

    pass\_login = Entry(login\_window,show = "\*")#show is used here because when the user enters their password it will come up with \*\*\* instead of the password, makes it look more authentic and secure

    pass\_login.grid(row = 1, column = 1)

    enter\_button = Button(login\_window,text="login", command=lambda: authentication(user\_login, pass\_login,login\_window),bg="#da001c") #the use of lambda here is to create a anonymous function that takes no arguements to then call the function authentication function, this is because in order to use the command option for the button i had to pass a function but for authentication i also need arguements for it therefore using lambda made it so i didnt need to pass arguments

    enter\_button.grid(columnspan=3)

    login\_window.mainloop()

    return user\_login, pass\_login,login\_window #returns the username and password from the login\_window to be used in the authentication function

def decide\_winner():

        update\_score()

        if p1\_card\_total > p2\_card\_total:

            messagebox.showinfo("player 1 wins!!", #i used these notification boxes to show that the game was over and requiring the players to finish the game, it also looked better than just having text in the gui

                f"player 1 wins with {p1\_card\_total} cards")

            winner = p1\_card\_total # winner score is amount of cards the winning player (e.g. player 1) had

        elif p1\_card\_total < p2\_card\_total:

            messagebox.showinfo("player 2 wins",

                f"player 2 wins with {p2\_card\_total} card ")

            winner = p2\_card\_total

        else:

            messagebox.showinfo("It's a tie!","Its a tie!")

            winner = p1\_card\_total

        score\_board.destroy()#this will close the scoreboard window signifying that the game is over

        window\_to\_ask\_for\_name(winner) #this will pass the winners score to the function to be entered into the leaderboard

def rules\_section():

    from os import startfile

    startfile("rules.txt") # this will open the notepad app to open the rules.txt file

def main\_menu():

    main\_menu\_window = Tk()

    main\_menu\_window.configure(background="light blue")

    main\_menu\_window.title("Main Menu")

    main\_menu\_window.geometry("300x200")

    font\_for\_game = Font(family="Segoe UI Black",size=12,weight="bold") #this is used for getting the type of font, letter size, and making the letters bold for the title of the window

    welcome\_tag = Label(main\_menu\_window,text = "Welcome to the card game",font=font\_for\_game,fg="green",bg="light blue")

    welcome\_tag.pack()

    play\_button = Button(main\_menu\_window,text = "Draw Cards", command= draw\_cards)#when this buttn is pressed, the draw\_cards function is called, starting the game

    play\_button.pack()

    leaderboard\_button = Button(main\_menu\_window,text="Leaderboard", command=leaderboard)#this is display the leaderboard

    leaderboard\_button.pack()

    rules\_button = Button(main\_menu\_window,text= "How to play", command=rules\_section)#this will open the rules

    rules\_button.pack()

    quit\_button = Button(main\_menu\_window,text="Quit", command=quit)#this will stop the program and close it

    quit\_button.pack()

    main\_menu\_window.update()

    def change\_colour(): #this function is used to change the colour of the "draw cards" button, making it more appealing and less boring

        colour = "#{0:06x}".format(r.randint(0,256\*\*3-1))#this changes the colour of the button randomly

        play\_button.config(bg=colour)#this will update the colour of the button

        play\_button.after(200,change\_colour)#thus will change the colour of the button every 200 milliseconds

    change\_colour()

def update\_score():

    p1\_score\_label.config(text=p1\_card\_total) #this will show will update the score of the players after each round

    p2\_score\_label.config(text=p2\_card\_total)

    score\_board.update()

def window\_to\_ask\_for\_name(winner):

    name\_window = Tk()

    name\_window.geometry("600x100")

    name\_window.configure(background="white")

    ask\_name = Label(name\_window,text="Enter your name to be added to the leaderboard (MAX LETTERS 5)",font = font\_for\_scorelabels,fg="purple",bg="white")

    enter\_name = Entry(name\_window)

    ask\_name.pack()

    enter\_name.pack()

    def add\_to\_leaderboard():#the reason for a function inside the function is because it would reduce the amount of lines needed in this code, also no need to make any variables global or return them

        name = enter\_name.get()

        length = len(name)

        if length <= 5:

            update\_leaderboard(winner,name)

            name\_window.destroy()

        elif length > 5 or length == 0:

            messagebox.showerror("Error","Name is too long, please try again") #error checking which alerts the user that their name is too long and gives them another chance to enter or shorten a names

    name\_button = Button(name\_window,text="Add to leaderboard",command=add\_to\_leaderboard,fg="purple") # this will call the add\_leaderboard

    name\_button.pack()

    name\_window.mainloop()

def update\_leaderboard(winner,name): #this function opens the leaderboard.csv file and write the name and score of the winner of the game

    file = open("Leaderboard.csv","a")

    file.write(f"{name},{winner}\n")

    file.close()

def leaderboard():

    import csv #this is needed to read the csv file as python cant normally read these

    x = 1

    leaderboard\_window = Tk() #creating a seperate window to display the leaderboard

    leaderboard\_window.geometry = ("150x100")

    leaderboard\_window.title = "Leaderboard"

    with open("Leaderboard.csv", "r") as file:

        reader = csv.reader(file)

        next(reader)  #skips the header row

        sorted\_data = sorted(reader, key=lambda row: int(row[1]), reverse=True)#lambda is used here to sort the data in based on their score in descending order as if lambda wasnt used the sorted function will automatically do it in ascending order not based on what on the order of data

    listbox = Listbox(leaderboard\_window)#creates a list box to display the leaderboard

    listbox.insert(0,"Pos Name Score")

    for index, data in enumerate(sorted\_data[:5]):

        listbox.insert(x,f"{index+1} {data[0]} {data[1]}")

        x = x + 1#this is the position of the player in the leaderboard

    listbox.pack()

    leaderboard\_window.mainloop()

#===========================================================BACK-END=========================================================================================#

#username = john password = test\_123

def authentication(user\_login, pass\_login,login\_window):

    db = mysql.connector.connect( # use of mysql to store usernames and password as i thought it would be better than just using a csv file or txt file

        host="localhost",

        user="root",

        password="root",

        database="store\_information" # these are needed for the python program to connect to the mysql database

    )

    mycursor = db.cursor()

    username = user\_login.get() #gets username from the login window

    passcode = pass\_login.get() #gets password from the login window

    sql\_query = "SELECT \* FROM info WHERE username = %s AND password = %s" #selects everything from the info table where the username = username entered and the password = password entered

    mycursor.execute(sql\_query, (username, passcode)) # executes the query

    result = mycursor.fetchall()

    if result: #if a result is returned from the query then then the user is allowed to login (only one username and pasasword on this whole database)

        messagebox.showinfo("Login successful", f"Welcome back {username}!")

        main\_menu() #opens the main menu

        login\_window.destroy() #closes the login window

    else:

        messagebox.showerror("Login failed", "Incorrect username or password, please try again.")#if thw username or password are false and there is no match for the username or password then this error message will be displayed

def draw\_cards():

    try:

        global p1\_card, p2\_card

        p1\_card = [i for i in deck[0]] # takes the first card from the deck

        deck.pop(0) # removes the first card from the deck

        p2\_card = [i for i in deck[0]] # takes the second card from the deck

        deck.pop(0) # removes the second card from the deck and so on...

        #i found that using a dictionaryhere was better then a using an if-statement as using if-statemnets here made my code ridiculously long and inefficient

        card\_images = [ #i used a dictionary here so instead of using a large amount of if-statements to find what card was drawn, the program can just go through the dictionary and compare it with the cards drawn to then output a image of the card drawn

        {"colour": "red", "number": 1, "img": "red 1.png"},

        {"colour": "red", "number": 2, "img": "red 2.png"},

        {"colour": "red", "number": 3, "img": "red 3.png"},

        {"colour": "red", "number": 4, "img": "red 4.png"},

        {"colour": "red", "number": 5, "img": "red 5.png"},

        {"colour": "red", "number": 6, "img": "red 6.png"},

        {"colour": "red", "number": 7, "img": "red 7.png"},

        {"colour": "red", "number": 8, "img": "red 8.png"},

        {"colour": "red", "number": 9, "img": "red 9.png"},

        {"colour": "red", "number": 10, "img": "red 10.png"},

        {"colour": "yellow", "number": 1, "img": "yellow 1.png"},

        {"colour": "yellow", "number": 2, "img": "yellow 2.png"},

        {"colour": "yellow", "number": 3, "img": "yellow 3.png"},

        {"colour": "yellow", "number": 4, "img": "yellow 4.png"},

        {"colour": "yellow", "number": 5, "img": "yellow 5.png"},

        {"colour": "yellow", "number": 6, "img": "yellow 6.png"},

        {"colour": "yellow", "number": 7, "img": "yellow 7.png"},

        {"colour": "yellow", "number": 8, "img": "yellow 8.png"},

        {"colour": "yellow", "number": 9, "img": "yellow 9.png"},

        {"colour": "yellow", "number": 10, "img": "yellow 10.png"},

        {"colour": "black", "number": 1, "img": "black 1.png"},

        {"colour": "black", "number": 2, "img": "black 2.png"},

        {"colour": "black", "number": 3, "img": "black 3.png"},

        {"colour": "black", "number": 4, "img": "black 4.png"},

        {"colour": "black", "number": 5, "img": "black 5.png"},

        {"colour": "black", "number": 6, "img": "black 6.png"},

        {"colour": "black", "number": 7, "img": "black 7.png"},

        {"colour": "black", "number": 8, "img": "black 8.png"},

        {"colour": "black", "number": 9, "img": "black 9.png"},

        {"colour": "black", "number": 10, "img": "black 10.png"},

    ]

        for card in card\_images: #loop through the dictionary

            if card["colour"] == p1\_card[0] and card["number"] == p1\_card[1]: #compares the colour of the card to the colours in the dictionary + the numbers

                p1\_pic\_card = Image.open(card["img"]) #once the colour and number is found then a image corresponding to the colour and the number will be outputed to the score\_board

                photo = ImageTk.PhotoImage(p1\_pic\_card)

                p1\_display\_drawn\_card.config(image=photo)

                p1\_display\_drawn\_card.image = photo #this is required as without it the images will appear blank

                break

        for card in card\_images:

            if card["colour"] == p2\_card[0] and card["number"] == p2\_card[1]:

                p2\_pic\_card = Image.open(card["img"])

                photo = ImageTk.PhotoImage(p2\_pic\_card)

                p2\_display\_drawn\_card.config(image=photo)

                p2\_display\_drawn\_card.image = photo

                break

        cards\_drawn.config(text=f"Player 1 drew {p1\_card} and player 2 drew {p2\_card}")

        compare\_cards(p1\_card, p2\_card)

    except IndexError:#i used a try and except here as a way of error checking as when there are no more cards and the game is finished, the "draw cards" button can still be clicked causing a possible crash

                    #therefore by using try and except and identify the which error occurred i can display and error message stating why theres a error and if they want to play again they would have to restart the program

        messagebox.showerror("Error","There are no more cards, to play again restart the game")

def compare\_cards(p1\_card, p2\_card):

    global p1\_card\_total, p2\_card\_total

    if p1\_card[0] == p2\_card[0]: #if the players have a card of the same colour then the program will compare their numbers

        if p1\_card[1] > p2\_card[1]:

            winner\_of\_round.config(text="👑Player 1 wins!!!👑",font=font\_for\_scorelabels)

            p1\_card\_total += 2 #player who wins will gain 2 points 1 for their card and the another for taking the other players card

            p1\_winning\_cards.append(p1\_card) #adds the cards used in this round to this variable

            p1\_winning\_cards.append(p2\_card)

        elif p1\_card[1] < p2\_card[1]:

            winner\_of\_round.config(text="👑Player 2 wins!!!👑",font=font\_for\_scorelabels)

            p2\_card\_total += 2

            p2\_winning\_cards.append(p2\_card)

            p2\_winning\_cards.append(p1\_card)

    else:

        if (p1\_card[0] == "red" and p2\_card[0] == "black") or (p1\_card[0] == "black" and p2\_card[0] == "yellow") or (p1\_card[0] == "yellow" and p2\_card[0] == "red"): #compares the different colour combinations there are

            winner\_of\_round.config(text="👑Player 1 wins!!!👑",font=font\_for\_scorelabels)

            p1\_card\_total += 2

            p1\_winning\_cards.append(p1\_card)

            p1\_winning\_cards.append(p2\_card)

        else:

            winner\_of\_round.config(text="👑Player 2 wins!!!👑",font=font\_for\_scorelabels)

            p2\_card\_total += 2

            p2\_winning\_cards.append(p2\_card)

            p2\_winning\_cards.append(p1\_card)

    next\_round()#opens the function if the deck is empty

    update\_score() #this opens the function to update the score after each round

def next\_round():

    if len(deck) == 0: # whden there are no more cards in the deck, game ends and winner is decided in the decide\_winner function

        decide\_winner()

#main program also back end

#this is here as i needed to make most of variables/windows here global variables as if they werent then the game wouldnt work as well as it currently is, this does however result in the score board being displayed with the login screen

p1\_card\_total = 0 # variable of the current score of each player

p2\_card\_total = 0

#score board in main program

score\_board = Tk()

score\_board.title("Current Score")

score\_board.geometry("500x400")

font\_for\_scorelabels = Font(family = "Segoe UI Black",size=12,weight="bold")

score\_board.configure(background="grey")

p1\_title = Label(score\_board,text="player 1 score",font=font\_for\_scorelabels,bg="grey",fg="#ff0001") #shows the title for players score

p1\_score\_label = Label(score\_board,text=p1\_card\_total,bg="grey",font=font\_for\_scorelabels,fg="#ff0001") #shows the current score of the players

p2\_title = Label(score\_board,text="player 2 score",font=font\_for\_scorelabels,bg="grey",fg="#0000ff")

p2\_score\_label = Label(score\_board,text=p2\_card\_total,bg="grey",font=font\_for\_scorelabels,fg="#0000ff")

cards\_drawn = Label(score\_board,text="",bg="grey",fg="black",font=font\_for\_scorelabels)

p1\_display\_drawn\_card = Label(bg="grey") #this is where the images of the current drawn cards will be

p2\_display\_drawn\_card = Label(bg="grey")

winner\_of\_round = Label(bg="grey",fg="orange")

p1\_title.pack()

p1\_score\_label.pack()

p2\_title.pack()

p2\_score\_label.pack()

winner\_of\_round.pack()

cards\_drawn.pack()

p1\_display\_drawn\_card.place(x=100,y=200) #this is the position of where the images of the current drawn cards will be displayed

p2\_display\_drawn\_card.place(x=350,y=200)

p1\_winning\_cards = [] #list to hold the winning cards of the players

p2\_winning\_cards = []

colours = ["red", "yellow", "black"] #list of the differnt colours

numbers = range(1, 11)#this chooses a random number between 1 and 11

deck = [(colour, number) for colour in colours for number in numbers] #this inputs a tuple into a list with a randomized number and colour

r.shuffle(deck) # this shuffles the deck

login\_screen() # this opens the login screen

TESTING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| test | Reason for test | Expected result | Actual result | Video |
| Clicking the draw button when the game is over | Because some players may not be paying attention and may click the button again to draw more cards | An error mess popping up saying that there are no more cards | Error message popped up saying there isn’t anymore cards |  |
| Entering a name longer than 5 characters into the leaderboard | Someone may have a name longer than 5 characters so they will try and enter that into the leaderboard | Error message saying that they went over the limit | Error message came up saying the name is too long |  |
| Closing the score board before starting the game | Someone may think that the game is broken as the scoreboard open before the game actually started | The game will crash | An tkinter error came up |  |