(Established under Karnataka Act No. 16 of 2013) 100 Feet Ring Road,
BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Arcade World

PES2UG21CS333- Nihal T M
PES2UG21CS447- S L Medha
PES2UG21CS609-Vishal S

Arcade World

Table of contents:

1) Modules Imported

2) Introduction

3) Abstract

4) Design-Code

5) Result

6) Future Enhancement

1) Modules Imported:

a) tkinter-It is an in-built module used to take care of GUI part of our Arcade World.

b) pygame-It is an in-built module used to make the Snake and Apple game.

c) turtle-It is an in-built module used to make our Wordle and Ping-Pong game.

d) time-It is an in-built module we used to make “movement” games progressively harder.

e) PIL-It is an in-built module we used to add Images to give the user the best Gaming Experience.

f) random-It is an in-built module we used to in User vs Computer games.

g) mailing-It is a user defined module we created by using the in-built module smtplib to manage all our mailing components to provide the smoothest and safest Register and Login for the user.

2) Introduction:

The name of our Python Project is Arcade World. As the name suggests, our project is a game selection window with five games namely Tic-Tac-Toe, Rock-Paper-Scissors, Wordle, Ping-Pong, and Snake&Apple. We have implemented various modules to bring our project to what it is now. Our project will hit gamers right in nostalgia. The 1990s were an incredible era for gaming. For those who experienced all it had to offer, this project will surely bring back those good memories. Our games are also made in a way such that there are minimal to almost no errors while gaming.

3) Abstract:

The goal of the project was to create a single game Tic-Tac-Toe but as we built the game we experienced joy and motivation to build more and more and kept adding games until we ended up making 5 games. Throughout our project making journey we asked ourselves this question- Would a real-world user like to play the game we built? This question kept popping in our head and we kept adding new features, new games and seamless switching between games. We also ended up picking up a lot of skills like team work, communication and of course some really good code.

4) Design Code:

mailing.py:

```
import random

import smtplib

from email.message import EmailMessage

import imghdr

from rand_pass_gen import generate_random_password,generate_2FA_code


EMAIL_ADDRESS='nihaltmdev@gmail.com'

EMAIL_PASSWORD='NihalwritesCode'


def welcome(a,b):

    global EMAIL_ADDRESS,EMAIL_PASSWORD,msg

    msg=EmailMessage()

    msg['Subject']='Welcome to Tic-Tac-Toe'

    msg['From']=EMAIL_ADDRESS

    msg['To']=a

    twofa=generate_2FA_code()

     msg.set_content(f"'Dear {b},\nRelive your childhood with a game that will give make you feel nostalgic and experience euphoria!

\nEnjoy the Game\nHere is your Confirmation code- {twofa}\nWith Warm Regards,\nNMV Team.'")

    with smtplib.SMTP_SSL('smtp.gmail.com',465) as smtp:
```

```
        smtp.login(EMAIL_ADDRESS,EMAIL_PASSWORD)
        smtp.send_message(msg)
        return twofa


def send_pass(a,b,c):
    global EMAIL_ADDRESS,EMAIL_PASSWORD,msg
    msg=EmailMessage()
    msg['Subject']='Tic Tac Toe password Reset'
    msg['From']=EMAIL_ADDRESS
    msg['To']=a
    new_pass=c
        msg.set_content(f'''Dear {b},\nWe realised that you wanted to login to your favourite game but could not as you had forgotten the password.
\nSo here is your password- {new_pass}\nPlease do not forget to delete this email once you have logged in\nEnjoy the Game\nWith Warm Regards,\nNMV Team.''')
    with smtplib.SMTP_SSL('smtp.gmail.com',465) as smtp:
        smtp.login(EMAIL_ADDRESS,EMAIL_PASSWORD)
        smtp.send_message(msg)
    return new_pass


def two_FA(a,b):
    global EMAIL_ADDRESS,EMAIL_PASSWORD,msg
    msg=EmailMessage()
```

```
    msg['Subject']='Tic Tac Toe 2FA'
    msg['From']=EMAIL_ADDRESS
    msg['To']=a
    twofa=generate_2FA_code()
        msg.set_content(f'"Dear {b},\nWe realised that you wanted to login to your favourite game.
\nSo here is the 2FA code- {twofa}\nEnjoy the Game\nWith Warm Regards, \nNMV Team.'")
    with smtplib.SMTP_SSL('smtp.gmail.com',465) as smtp:
            smtp.login(EMAIL_ADDRESS,EMAIL_PASSWORD)
            smtp.send_message(msg)
    return twofa
```

rand_pass_gen.py

```
import string
import random


characters = list(string.ascii_letters + string.digits + "!@#$%^&*()")
numbers=["0","1","2","3","4","5","6","7","8","9"]


def generate_random_password():
        length=10
        random.shuffle(characters)
```

```
        password = []
        for i in range(length):
                password.append(random.choice(characters))
        random.shuffle(password)
        return "".join(password)


def generate_2FA_code():
        length=6
        random.shuffle(numbers)
        twofa=[]
        for i in range(length):
                twofa.append(random.choice(numbers))
        random.shuffle(twofa)
        return "".join(twofa)
```

```
from tkinter import *
from tkinter.messagebox import showinfo,askyesno
from tkinter.simpledialog import askstring
import mailing as em
from PIL import Image,ImageTk
import random,turtle
import pygame
from pygame.locals import *
import time


def Gameon(back_buttonIMG):
    frame2.pack_forget()
    RorL.title("Tic-Tac-Toe Game")
    numbers=[1,2,3,4,5,6,7,8,9]
    # y='X' for player1 and 'O' for player2
    y=""
    # x is the counter to keep counting the number of chances
    x=0
    #boards is a list to store the mark with respect to the cell number
    boards=[""]*10
    counter1=0
    counter2=0
```

```
def reset():
    nonlocal boards,numbers,x,y,counter1,counter2
    x=0
    y=""
    boards=[""]*10
    numbers=[1,2,3,4,5,6,7,8,9]
    counterlabel1.config(text=f"Player 1 Score:{counter1}")
    counterlabel2.config(text=f"Player 2 Score:{counter2}")
    for i in numbers:
        lb[i-1].config(text="")


def result(boards,mark):
    return ((boards[1] == boards[2] == boards [3] == mark)
            or (boards[4] == boards[5] == boards [6] == mark)
            or (boards[7] == boards[8] == boards [9] == mark)
            or (boards[1] == boards[4] == boards [7] == mark)
            or (boards[2] == boards[5] == boards [8] == mark)
            or (boards[3] == boards[6] == boards [9] == mark)
            or (boards[1] == boards[5] == boards [9] == mark)
            or (boards[3] == boards[5] == boards [7] == mark))
```

```
    def decider(a,b,n):
        nonlocal x,y,numbers,counter1,counter2
        x,y,num=a,b,n
        if x%2==0:
            y='X'
            boards[num]=y
        elif x%2!=0:
            y='O'
            boards[num]=y
            lb[num-1].config(text=y,font="times 10") #Display X or O on the button clicked by user
        x=x+1
        mark=y
         # Here we are calling the result() to decide whether we have got the winner or not
        if(result(boards,mark) and mark=='X' ):
            #If Player1 is the winner show a dialog box stating the winner
            showinfo("Result","Player 1 wins")
            counter1+=1
            #Call the destroy function to close the GUI
            reset()
        elif(result(boards,mark) and mark=='O'):
            showinfo("Result","Player 2 wins")
```

```
            counter2+=1
            reset()

    def click(num):
        nonlocal x,y,numbers
        """ To Check which button has been clicked to avoid over-writing"""
        if num in numbers:
            numbers.remove(num)
            decider(x,y,num)

        # If we have not got any winner, display match has been tied.
        if(x>8 and result(boards,'X')==False and result(boards,'O')==False):
            showinfo("Result","Match Tied")
            reset()


backbutton=Button(frame3,image=back_buttonIMG,command=lambda:f3_to_f2()
)
    backbutton.grid(row=1,column=1)
    counterlabel1=Label(frame3,text="Player 1 Score:0")
    counterlabel1.grid(row=2,column=1)
    counterlabel2=Label(frame3,text="Player 2 Score:0")
    counterlabel2.grid(row=3,column=1)
```

```
l1=Label(frame3,text=" player1 : X",font="times 18")
l1.grid(row=0,column=1)
l2=Label(frame3,text=" player2 : O",font="times 18")
l2.grid(row=0,column=2)
 # combined building buttons and gridding the button
b1=Button(frame3,width=24,height=12,command=lambda:click(1))
b1.grid(row=1,column=6)
b2=Button(frame3,width=24,height=12,command=lambda:click(2))
b2.grid(row=1,column=7)
b3=Button(frame3,width=24,height=12,command=lambda: click(3))
b3.grid(row=1,column=8)
b4=Button(frame3,width=24,height=12,command=lambda: click(4))
b4.grid(row=2,column=6)
b5=Button(frame3,width=24,height=12,command=lambda: click(5))
b5.grid(row=2,column=7)
b6=Button(frame3,width=24,height=12,command=lambda: click(6))
b6.grid(row=2,column=8)
b7=Button(frame3,width=24,height=12,command=lambda: click(7))
b7.grid(row=3,column=6)
b8=Button(frame3,width=24,height=12,command=lambda: click(8))
b8.grid(row=3,column=7)
b9=Button(frame3,width=24,height=12,command=lambda: click(9))
```

```
    b9.grid(row=3,column=8)
    lb=[b1,b2,b3,b4,b5,b6,b7,b8,b9] #Storing buttons in a list to reduce code size


    frame3.pack(expand=1)


def registration(a,b,c,d):
    f=open("UserDataBase.txt")
    content=(f.read().splitlines())
    fw=open("UserDataBase.txt","a")
    for m in content:
        x=m.split(",")
        if c!=d:
            showinfo("Incorrect Input","Password and Confirm Password is different!")
            break
        if a==x[1]:
                    showinfo("Alert!","An account using the provided email already exists\n\tYou can try logging in!")
            break
        else:
            authentication=em.welcome(a,b)
              code=askstring("Mail confirmation","Please enter confirmation code that was mailed to you")
            if code==authentication:
```

```
            response=askyesno("Real Quick!","Would you like to enable 2FA and make your account more secure?")
            if response=="Yes":
                s="\n"+b+","+a+","+c+",2FA"
                fw.write(s)
                fw.close()
                f2()
                RorL.title("Game Selection")
                break
            else:
                s="\n"+b+","+a+","+c
                fw.write(s)
                fw.close()
                f2()
                RorL.title("Game Selection")
                break


def Login(a,b):
    f=open("UserDataBase.txt","r")
    content=(f.read()).splitlines()
    c=1
    for m in content:
```

```
        x=m.split(",")
        if a==x[1]:
            c=0
            if b==x[2]:
                if "2FA" in m:
                    authentication=em.two_FA(a,x[0])
                     code=askstring("2FA confirmation","Please enter the 6-digit 2FA code that was mailed to you")
                    if code==authentication:
                        f2()
                        RorL.title("Game Selection")
                        break
                else:
                    f2()
                    RorL.title("Game Selection")
                    break
            else:
                showinfo("Wrong password","The password you have entered is wrong")
                break
    if c!=0:
             showinfo("Wrong email","The email you provided hasn't registered an account\n\tYou can try registering a new account!")
```

```
def fpass():
    a=askstring("Forgot Password","Please enter your email address")
    f=open("UserDataBase.txt","r")
    content=(f.read()).splitlines()
    for m in content:
        x=m.split(",")
        if a==x[1]:
            c=0
            em.send_pass(a,x[0],x[2])
                showinfo("Alert!","The password has been sent to your email,\nplease login and delete the email thereafter ")
    if c!=0:
                showinfo("Wrong Email","There is no Game account in the mail you specified!")


def pong():
    pygame.mixer.init()
    pygame.mixer.music.load('Sadches - Retro arcade.mp3')
    pygame.mixer.music.play()


    # Functions
    def paddle_a_up():
        global y
```

```
    y = paddle_a.ycor()
    y += 20
    paddle_a.sety(y)

def paddle_a_down():
    global y
    y = paddle_a.ycor()
    y -= 20
    paddle_a.sety(y)

def paddle_b_up():
    global y
    y = paddle_b.ycor()
    y += 20
    paddle_b.sety(y)

def paddle_b_down():
    global y
    y = paddle_b.ycor()
    y -= 20
    paddle_b.sety(y)
```

```
def resetpong():
    global wn,score_a,score_b,x,y,paddle_a,paddle_b,tup,ball,pen
    wn = turtle.Screen()
    wn.title("Pong")
    wn.bgcolor("black")
    wn.setup(width=800, height=600)
    wn.tracer(0)
    # Score


    # Paddle A
    paddle_a = turtle.Turtle()
    paddle_a.speed(0)
    paddle_a.shape("square")
    paddle_a.color("white")
    paddle_a.shapesize(stretch_wid=5,stretch_len=1)
    paddle_a.penup()
    paddle_a.goto(-350, 0)


    # Paddle B
    paddle_b = turtle.Turtle()
    paddle_b.speed(0)
    paddle_b.shape("square")
```

```
paddle_b.color("white")
paddle_b.shapesize(stretch_wid=5,stretch_len=1)
paddle_b.penup()
paddle_b.goto(350, 0)

# Ball
ball = turtle.Turtle()
ball.speed(0)
ball.shape("circle")
ball.color("white")
ball.penup()
ball.goto(0, 0)
ball.dx = 2
ball.dy = 2

# Pen



#sleep timer, controls the speed of the ball.
global x
x=True
```

```
    def destroys():
        wn.bye()
        pygame.mixer.music.pause()
        global x
        x=False
    tup=0.01
    score_a = 0
    score_b = 0
    pen = turtle.Turtle()
    pen.speed(0)
    pen.shape("square")
    pen.color("white")
    pen.penup()
    pen.hideturtle()
    pen.goto(0, 260)
        pen.write("Player A: 0  Player B: 0", align="center", font=("Courier", 24,
"normal"))
    resetpong()
    # Keyboard bindings
    wn.listen()
    wn.onkeypress(paddle_a_up, "w")
    wn.onkeypress(paddle_a_down, "s")
```

```
wn.onkeypress(paddle_b_up, "Up")
wn.onkeypress(paddle_b_down, "Down")
wn.onkeypress(destroys,"space")
x=True
# Main game loop
while x:
    wn.update()

    tup-=0.000001
    time.sleep(tup)
    # Move the ball
    ball.setx(ball.xcor() + ball.dx)
    ball.sety(ball.ycor() + ball.dy)

    # Border checking

    # Top and bottom
    if ball.ycor() > 290:
        ball.sety(290)
        ball.dy *= -1

    elif ball.ycor() < -290:
```

```
        ball.sety(-290)
        ball.dy *= -1


    # Left and right
    if ball.xcor() > 350:
        score_a += 1
        pen.clear()
                pen.write("Player A: {}  Player B: {}".format(score_a, score_b), align="center", font=("Courier", 24, "normal"))
        ball.goto(0, 0)
        ball.dx *= -1


    elif ball.xcor() < -350:
        score_b += 1
        pen.clear()
                pen.write("Player A: {}  Player B: {}".format(score_a, score_b), align="center", font=("Courier", 24, "normal"))
        ball.goto(0, 0)
        ball.dx *= -1


    # Paddle and ball collisions
      if ball.xcor() < -340 and ball.ycor() < paddle_a.ycor() + 50 and ball.ycor() > paddle_a.ycor() - 50:
```

```
        ball.dx *= -1


    elif ball.xcor() > 340 and ball.ycor() < paddle_b.ycor() + 50 and ball.ycor() > paddle_b.ycor() - 50:
        ball.dx *= -1


def turtlegame():
    l=[0,1,2,3,4]
    word_list=['which', 'their', 'wipro', 'there', 'could']
    word_q=['Used in questions to ask type of a subject','Used to refer ownership','A popular company Harsham Premji','Used to refer location','Used when hypothetically thinking']
    answernumber = random.choice(l)# choose a random word or question from the list
    answer=word_list[answernumber]
    answerq=word_q[answernumber]
    y = 250 # y location
    def draw_square(x,y,col): # takes in x,y coordinates and a color
        turtle.penup()
        turtle.goto(x,y)
        turtle.pendown()
        turtle.fillcolor(col) # set the fillcolor
        turtle.begin_fill()     # start the filling color
```

```
    for i in range(4):     # drawing the square
        turtle.forward(50)
        turtle.right(90)
    turtle.end_fill() # ending the filling of the color


def input_guess(prompt):
    my_input = turtle.textinput("5 letter word", prompt)
     if my_input == None: return "       " # if you press cancel or submit with nothing
    elif len(my_input) != 5: return my_input[0:5] #sends the first five characters
    else: return my_input.lower() #sends lower case of the word. Means case does not make a difference.


def check_guess(my_input,answer,y):
    count = 0 #Need a count, because of for loop used
    x = -250 # x location
    for i in my_input:
        if i == answer[count]: draw_square(x,y,"green") #exact character match draws a green square
        elif i in answer: draw_square(x,y,"yellow") #else if character anywhere in word draws yellow
        else: draw_square(x,y,"red") # otherwise draws red
        count+=1 # add 1 to the count
```

```
        x += 75 # move x coordinate along by 75
    turtle.penup() #Moves the turtle penup
    turtle.goto(x,y-25) #Moves it slightly down for the word
        turtle.write(my_input,font=("Fixedsys", 15, "normal")) # font verdana, size 15, normal style


  for i in range(6): #Where the program starts
      guess_prompt = "What is guess "+str(i+1)+"?"+"\n"+answerq #Makes a nice string for the prompt
    my_input = input_guess(guess_prompt) #calls input_guess function
    check_guess(my_input,answer,y)  #checks the guess
    y -= 75 #y down by 75
    if my_input == answer:
        turtle.penup()
        turtle.goto(-300,-200) #Always draws the congratulations in the same place
        turtle.write("Well Done!",font=("Chiller", 60, "normal"))
        break
    else: #Only runs if the for loop executes completely. i.e. You've used all your guesses.
    turtle.penup()
    turtle.goto(-300,-200)
    turtle.write(answer,font=("Cooper Black", 42, "normal"))
  turtle.done() #Needs if you are using Pycharm and some other Python editors.
```

```
def rpsgame(back_buttonIMG):
    frame2.pack_forget()
    RorL.title('Rock Paper Scissor!!')

    title=Label(frame4,text='\n Rock Paper Scissor!! \n',fg='grey')
    title.config(font=('Courier',40))
    title.grid(row=0,column=2,columnspan=5)

backbutton=Button(frame4,image=back_buttonIMG,command=lambda:f4_to_f2()
)
    backbutton.grid(row=1,column=1)

    computer_input=""
    playercounter=0
    computercounter=0

    title2=Label(frame4,text='Enjoy the game.....\n \n',fg='green')
    title2.config(font=('Courier',20))
    title2.grid(row=1,column=3)

    options=Label(frame4,text='Options:\n',fg='orange')
```

```
    options.config(font=('Courier',18))
    options.grid(row=3,column=1)


    opt=['rock','paper','scissor']
    def winner(player_input):
                                                                    n o n l o c a l
your_score,computer_score,computer_input,playercounter,computercounter
        computer_input=random.choice(opt)


        if player_input==computer_input:
            title2.config(text="It is a Tie!")
            computer_choice.config(text="Computer's choice-->"+computer_input)
            your_choice.config(text="Your choice-->"+player_input)
                        elif (player_input=="rock" and computer_input=="paper") or
(player_input=="scissor" and computer_input=="rock") or (player_input=="paper"
and computer_input=="scissor"):
            title2.config(text="Computer won!")
            computercounter+=1
            computer_choice.config(text="Computer's choice-->"+computer_input)
            your_choice.config(text="Your choice-->"+player_input)
            computer_score.config(text="Computer's Score-->"+str(computercounter))
        else:
            title2.config(text="You win!")
```

```
            playercounter+=1
            your_score.config(text="Your Score-->"+str(playercounter))
            computer_choice.config(text="Computer's choice-->"+computer_input)
            your_choice.config(text="Player choice-->"+str(player_input))

    #buttons
    rock=Button(frame4,text='ROCK',width=15,command=lambda:winner("rock"))
    rock.grid(column=2,row=4)


paper=Button(frame4,text='PAPER',width=15,command=lambda:winner("paper"))
    paper.grid(row=4,column=3)


scissor=Button(frame4,text='SCISSOR',width=15,command=lambda:winner("sciss
or"))
    scissor.grid(row=4,column=4)

    your_choice=Label(frame4,text='Your choice-->(playername)')
    your_choice.grid(row=6, column=1)

    #results
    your_score=Label(frame4,text=f"Your Score-->{playercounter}")
```

```
    your_score.grid(row=6,column=3)


            computer_choice=Label(frame4,text=f"'Computer's choice-->{computer_input}'")
    computer_choice.grid(row=7,column=1)


            computer_score=Label(frame4,text=f"'Computer's Score-->{computercounter}'")
    computer_score.grid(row=7,column=3)
    frame4.pack()


def showpass(checkb,entry):
    if checkb.var.get():
        entry.config(show="")
    else:
        entry.config(show="*")


def SnakeGames():
    SIZE=40
    pause=True


    class Apple:
        def __init__(self,parent_screen):
```

```
        self.parent_screen=parent_screen
        self.image=pygame.image.load("apple.jpg").convert()
        self.x=120
        self.y=120

    def draw(self):
        self.parent_screen.blit(self.image,(self.x,self.y))
        pygame.display.flip()

    def move(self):
        self.x=random.randint(1,12)*SIZE
        self.y=random.randint(1,8)*SIZE


class Snake:
    def __init__(self,parent_screen,length):
        self.length=length
        self.parent_screen=parent_screen
        self.block=pygame.image.load("block.jpg").convert()
        self.x=[SIZE]*length
        self.y=[SIZE]*length
        self.direction="down"
```

```
def inc_length(self):
    self.length+=1
    self.x.append(-1)
    self.y.append(-1)

def draw(self):
    self.parent_screen.fill((110,110,5))
    for i in range(self.length):
        self.parent_screen.blit(self.block,(self.x[i],self.y[i]))
    pygame.display.flip()

def move_left(self):
    self.direction="left"

def move_right(self):
    self.direction="right"

def move_up(self):
    self.direction="up"

def move_down(self):
```

```
        self.direction="down"


    def walk(self):


        for i in range(self.length-1,0,-1):
            self.x[i]=self.x[i-1]
            self.y[i]=self.y[i-1]
        if self.direction=="up":
            self.y[0]-=40
        if self.direction=="down":
            self.y[0]+=40
        if self.direction=="right":
            self.x[0]+=40
        if self.direction=="left":
            self.x[0]-=40
        self.draw()



class Game:
    def __init__(self):
        pygame.init()
        self.surface=pygame.display.set_mode((750,500))
```

```
        self.surface.fill((110,110,5))
        self.snake=Snake(self.surface,1)
        self.snake.draw()
        self.apple=Apple(self.surface)
        self.apple.draw()
    #snake colliding with apple
    def is_collision(self,x1,y1,x2,y2):
        if x1>=x2 and x1<x2+SIZE:
            if y1>=y2 and y1<y2+SIZE:
                pygame.mixer.init()
                pygame.mixer.music.load('ding.mp3')
                pygame.mixer.music.play()
                return True

    def display_score(self):
        font=pygame.font.SysFont('arial',30)
        score=font.render(f"Score: {self.snake.length-1}",True,(255,255,255))
        self.surface.blit(score,(500,10))

    def play(self):
        self.snake.walk()
        self.apple.draw()
```

```
        self.display_score()
        pygame.display.flip()
        if self.is_collision(self.snake.x[0],self.snake.y[0],self.apple.x,self.apple.y):
            self.apple.move()
            self.snake.inc_length()

        for i in range(3,self.snake.length):
                                                                    if
self.is_collision(self.snake.x[0],self.snake.y[0],self.snake.x[i],self.snake.y[i]):
                pygame.mixer.init()
                pygame.mixer.music.load('crash.mp3')
                pygame.mixer.music.play()
                raise "Game Over"

    def show_game_over(self):
        self.surface.fill((110,110,5))
        font=pygame.font.SysFont('arial',30)
        line1=font.render(f"Game is Over, Your Score: {self.snake.length-1}",True,
(255,255,255))
        self.surface.blit(line1,(100,200))
          line2=font.render("To play again press Enter. To exit press Escape",True,
(255,255,255))
        self.surface.blit(line2,(100,250))
```

```
        pygame.display.flip()



    def run(self):
            running=True
            pause=False

            while running:
                for event in pygame.event.get():
                    if event.type==KEYDOWN:
                        if event.key==K_ESCAPE:
                            running=False

                        if event.key==K_RETURN:
                            pause=False
                            self.snake.length=1
                        if event.key==K_ESCAPE:
                            pause=True

                            self.show_game_over()
                        if event.key== K_UP:
```

```
                    self.snake.move_up()
                if event.key== K_DOWN:
                    self.snake.move_down()
                if event.key== K_LEFT:
                    self.snake.move_left()
                if event.key== K_RIGHT:
                    self.snake.move_right()

            elif event.type==QUIT:
                running=False

        try:
            if not pause:
                self.play()
        except Exception as e:
            self.show_game_over()
            pause=True
        time.sleep(0.25)
if __name__=="__main__":
    game=Game()
    game.run()
```

```
RorL=Tk()
RorL.title("Welcome to the Game terminal")
RorL.geometry("1500x1000")


frame5=Frame(RorL)


frame4=Frame(RorL)


frame3=Frame(RorL)


frame2=Frame(RorL)
bgIMG=ImageTk.PhotoImage(Image.open("GSelectionIMG.jpg"))
bglabel=Label(frame2,image=bgIMG)
bglabel.pack()
logoutIMG=ImageTk.PhotoImage(Image.open("LogoutIMG1.jpg"))
logoutbutton=Button(frame2,image=logoutIMG,bg="black",command=lambda:f2_
to_f1())
SelectLabel=Label(frame2,text="Choose a Game To Play:",font="times
20",bg="black",fg="Red")
btttIMG=ImageTk.PhotoImage(Image.open("TTTimage.png"))
back_buttonIMG=ImageTk.PhotoImage(Image.open("back_buttonIMG.jpg"))
```

```
bTTT=Button(frame2,image=btttIMG,width=200,height=200,command=lambda:Gameon(back_buttonIMG))

bTTT.place(x=50,y=120)

SelectLabel.place(x=550,y=50)

logoutbutton.place(x=1200,y=20)

rpsIMG=ImageTk.PhotoImage(Image.open("rpsIMG.png"))

rpsgamebutton=Button(frame2,image=rpsIMG,width=200,height=200,command=lambda:rpsgame(back_buttonIMG))

rpsgamebutton.place(x=300,y=120)

turtlegameIMG=ImageTk.PhotoImage(Image.open("turtle_gameIMG.jpg"))

turtlegamebutton=Button(frame2,image=turtlegameIMG,width=200,height=200,command=lambda:turtlegame())

turtlegamebutton.place(x=550,y=120)

PongIMG=ImageTk.PhotoImage(Image.open("pong_IMG.jpg"))

Ponggamebutton=Button(frame2,image=PongIMG,width=200,height=200,command=lambda:pong())

Ponggamebutton.place(x=800,y=120)

snakegameIMG=ImageTk.PhotoImage(Image.open("snakegame.png"))

SnakeGameButton=Button(frame2,image=snakegameIMG,width=200,height=200,command=lambda:SnakeGames())

SnakeGameButton.place(x=50,y=350)


frame1=Frame(RorL)

bgimg=ImageTk.PhotoImage(Image.open("RorLBGI.jpg"))
```

```
bglabel=Label(frame1,image=bgimg)
bglabel.pack()
rml=Label(frame1,text="Enter your email address:")
rnl=Label(frame1,text="Enter your name:")
rpl=Label(frame1,text="Enter your Password")
rcpl=Label(frame1,text="Confirm your Password")
rpe=Entry(frame1,width=40,show="*")
rcpe=Entry(frame1,width=40,show="*")
rme=Entry(frame1,width=40)
rne=Entry(frame1,width=40)
rml.place(x=400,y=200)
rme.place(x=400,y=225)
rnl.place(x=400,y=250)
rne.place(x=400,y=275)
rpl.place(x=400,y=300)
rpe.place(x=400,y=325)
rcpl.place(x=400,y=380)
rcpe.place(x=400,y=405)
lml=Label(frame1,text="Enter your email address:")
lml.place(x=750,y=200)
lme=Entry(frame1,width=40)
lme.place(x=750,y=225)
```

```
lpl=Label(frame1,text="Enter your Password")

lpl.place(x=750,y=250)

lpe=Entry(frame1,width=40,show="*")

lpe.place(x=750,y=275)

Registerbg=ImageTk.PhotoImage(Image.open("RegisterIMG.jpg"))

Loginbg=ImageTk.PhotoImage(Image.open("LoginIMG.jpg"))

rb=Label(frame1,image=Registerbg,bg="black")

lb=Label(frame1,image=Loginbg,bg="black")

register=Button(frame1,text="Register",width=10,height=2,bg="green",command=lambda:registration(rme.get(),rne.get(),rpe.get(),rcpe.get()) if rme.get()!="" and rne.get()!="" and rpe.get()!="" and rcpe.get()!="" else showinfo("Warning!","Please fill all fields!")) #mail#name#pass#confirmpass

login=Button(frame1,text="Login",width=10,height=2,bg="green",command=lambda:Login(lme.get(),lpe.get()) if lme.get()!="" and lpe.get()!="" else showinfo("Warning!","Please fill all fields!"))

forgotpass=Button(frame1,text="Forgot Password",width=12,height=1,bg="yellow",command=lambda:fpass())

forgotpass.place(x=900,y=300)

register.place(x=400,y=460)

login.place(x=750,y=330)

rb.place(x=400,y=100)

lb.place(x=750,y=100)


cbrpe,cbrcpe,cb=0,0,0
```

```
cbrpe=Checkbutton(frame1,text="Show Password",onvalue=True,offvalue=False,command=lambda:showpass(cbrpe,rpe))

cbrpe.var=BooleanVar(value=False)

cbrpe['variable']=cbrpe.var

cbrpe.place(x=400,y=350)

cbrcpe=Checkbutton(frame1,text="Show Password",onvalue=True,offvalue=False,command=lambda:showpass(cbrcpe,rcpe))

cbrcpe.var=BooleanVar(value=False)

cbrcpe['variable']=cbrcpe.var

cbrcpe.place(x=400,y=430)

cb=Checkbutton(frame1,text="Show Password",onvalue=True,offvalue=False,command=lambda:showpass(cb,lpe))

cb.var=BooleanVar(value=False)

cb['variable']=cb.var

cb.place(x=750,y=300)


def f4_to_f2():
    frame4.pack_forget()
    RorL.title("Game Selection")
    f2()


def f3_to_f2():
```

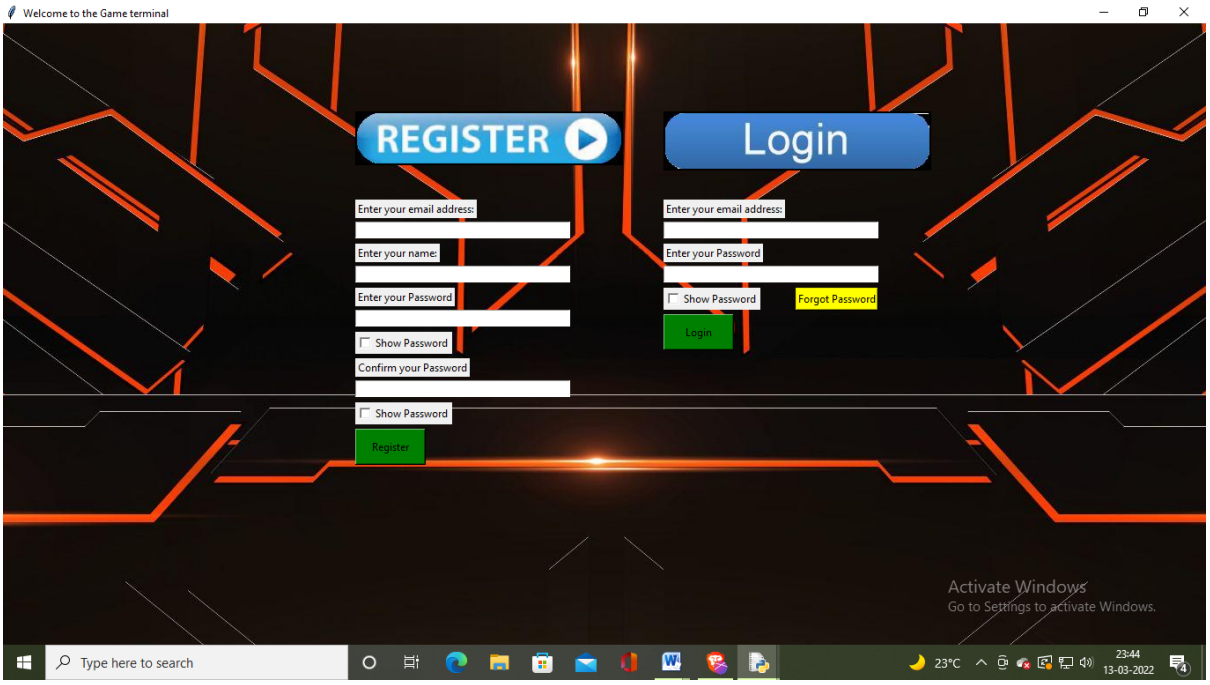
```
    frame3.pack_forget()
    RorL.title("Game Selection")
    f2()

def f2_to_f1():
    frame2.pack_forget()
    f1()

def f1():
    frame1.pack()

def f2():
    rme.delete(0,END)
    rne.delete(0,END)
    rpe.delete(0,END)
    rcpe.delete(0,END)

    lme.delete(0,END)
    lpe.delete(0,END)
    frame1.pack_forget()
    frame2.pack()
```

f1()

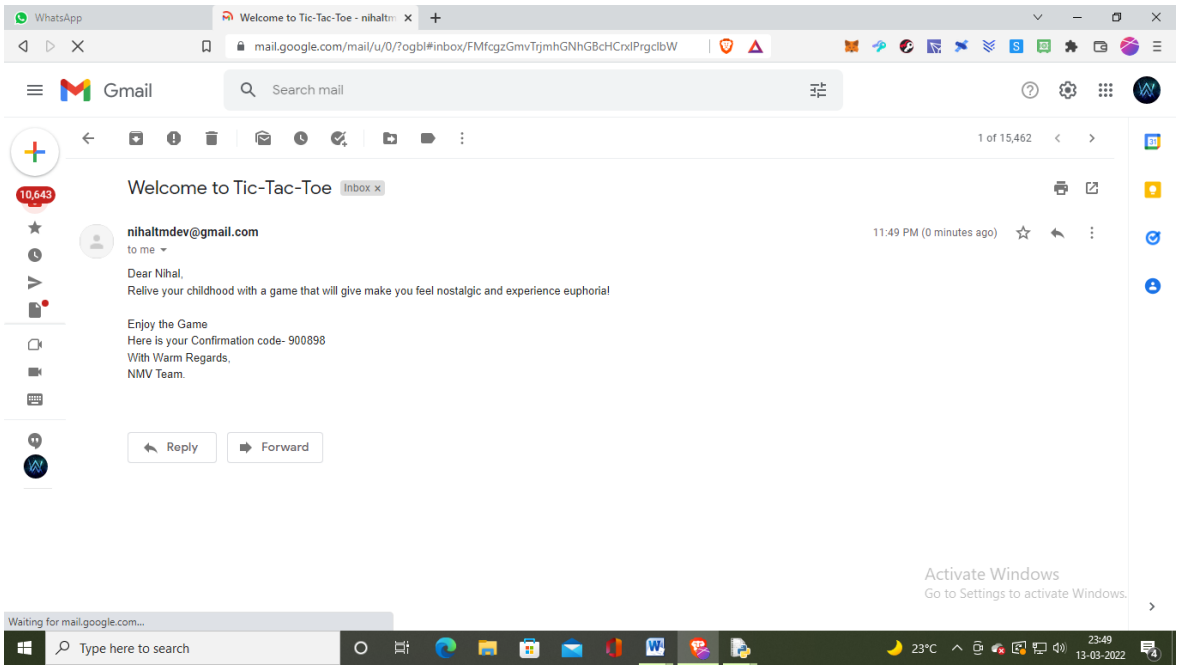
RorL.mainloop()

5) Result:

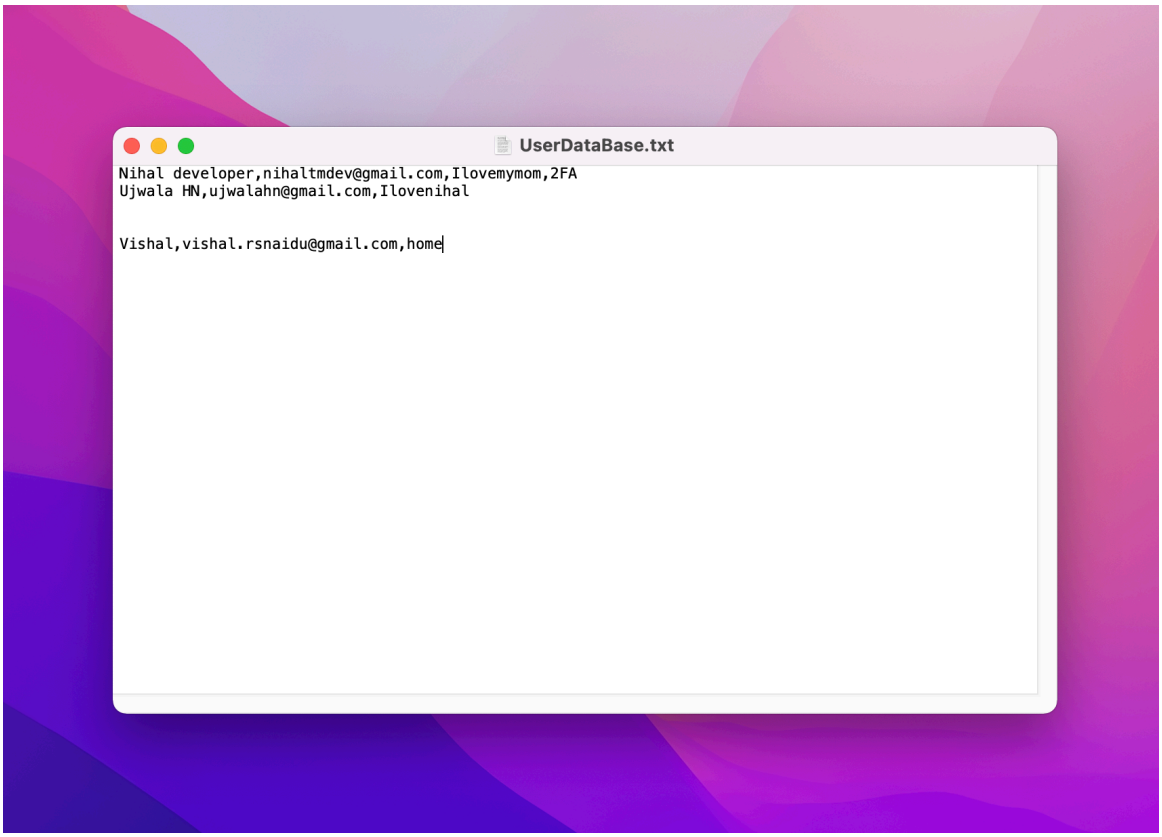
i) The first frame that requests the user to Register a new account if they are a new user or Login if they already are registered. (Information of Registered users is saved in a text file)



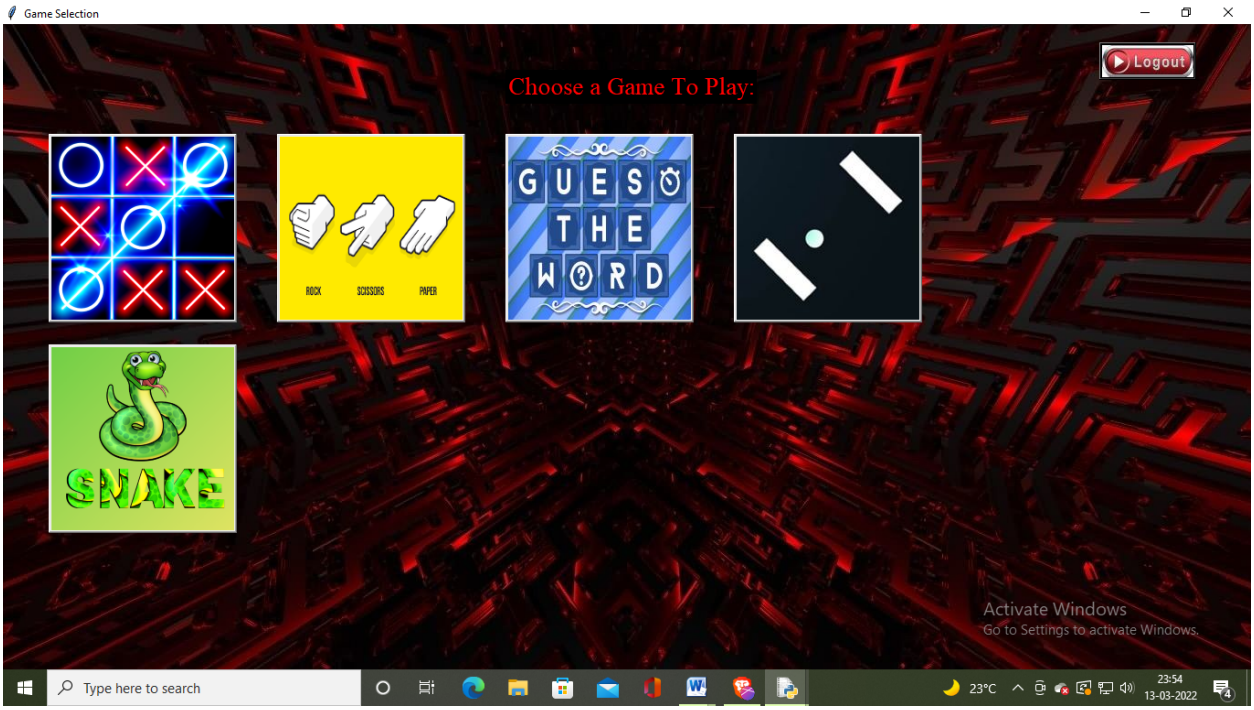
ii) **Mailing of randomly generated codes to protect user accounts:**



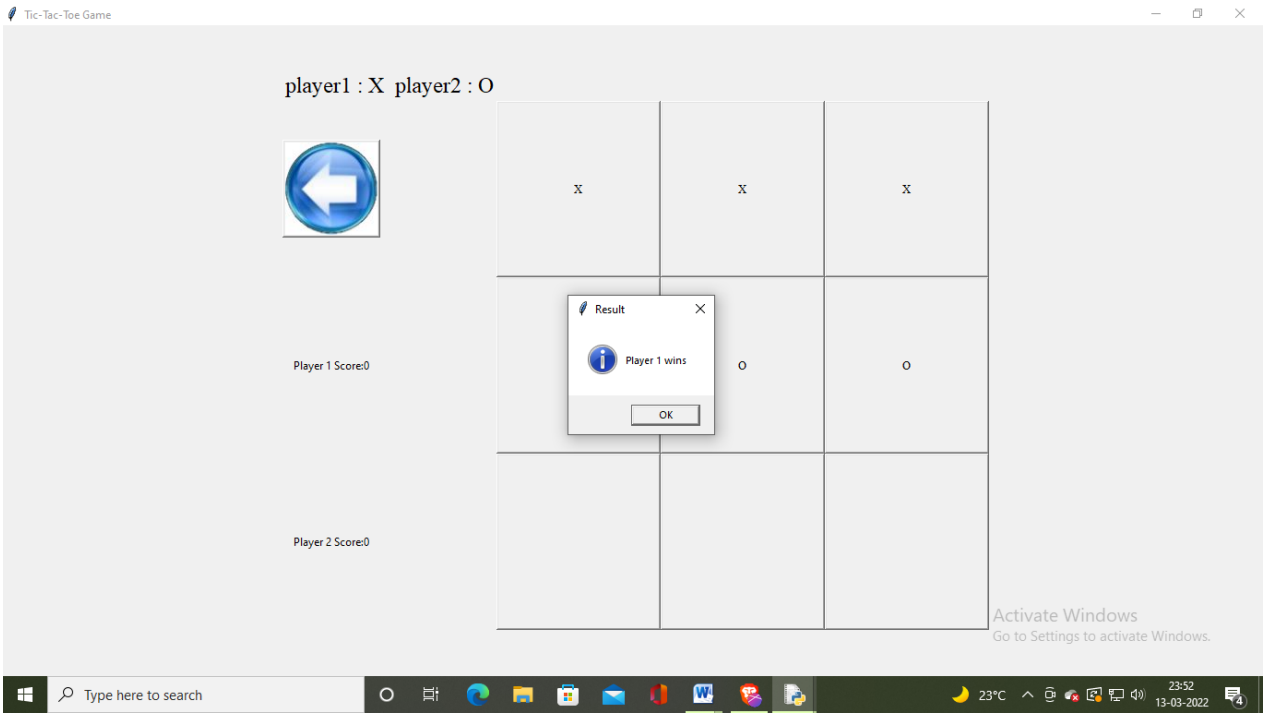
iii) **UserDataBase.txt which stores the registered accounts.**



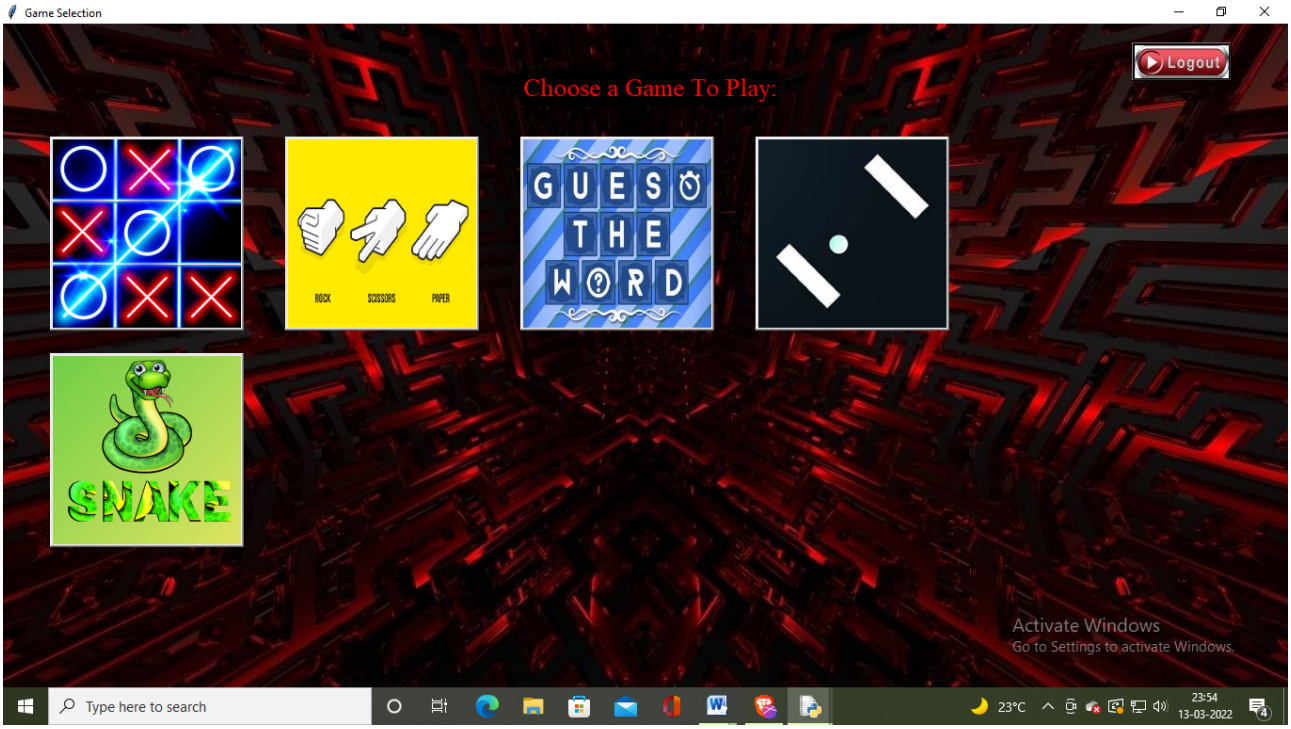
iv) Game Selection window to provide users a choice on which game to play!



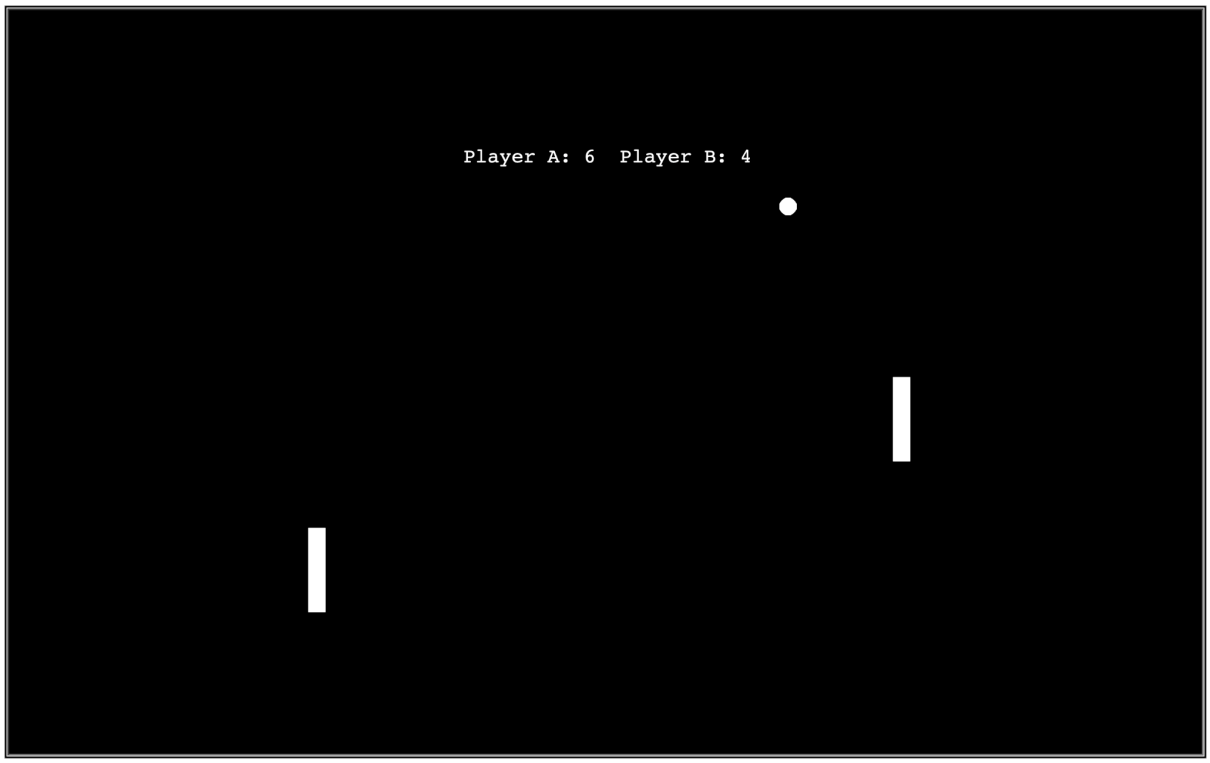
v) Player1 just won!



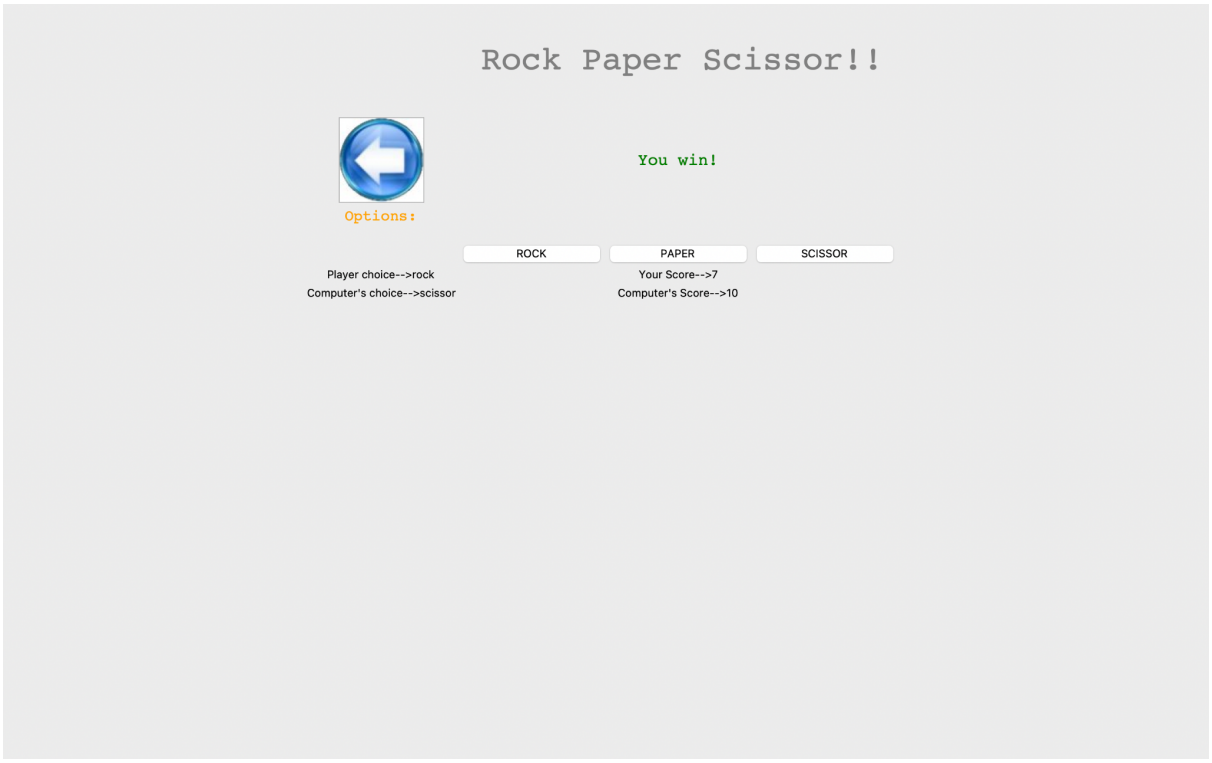
vi) **Want to play some other game? We got you covered! Just press the back button.**



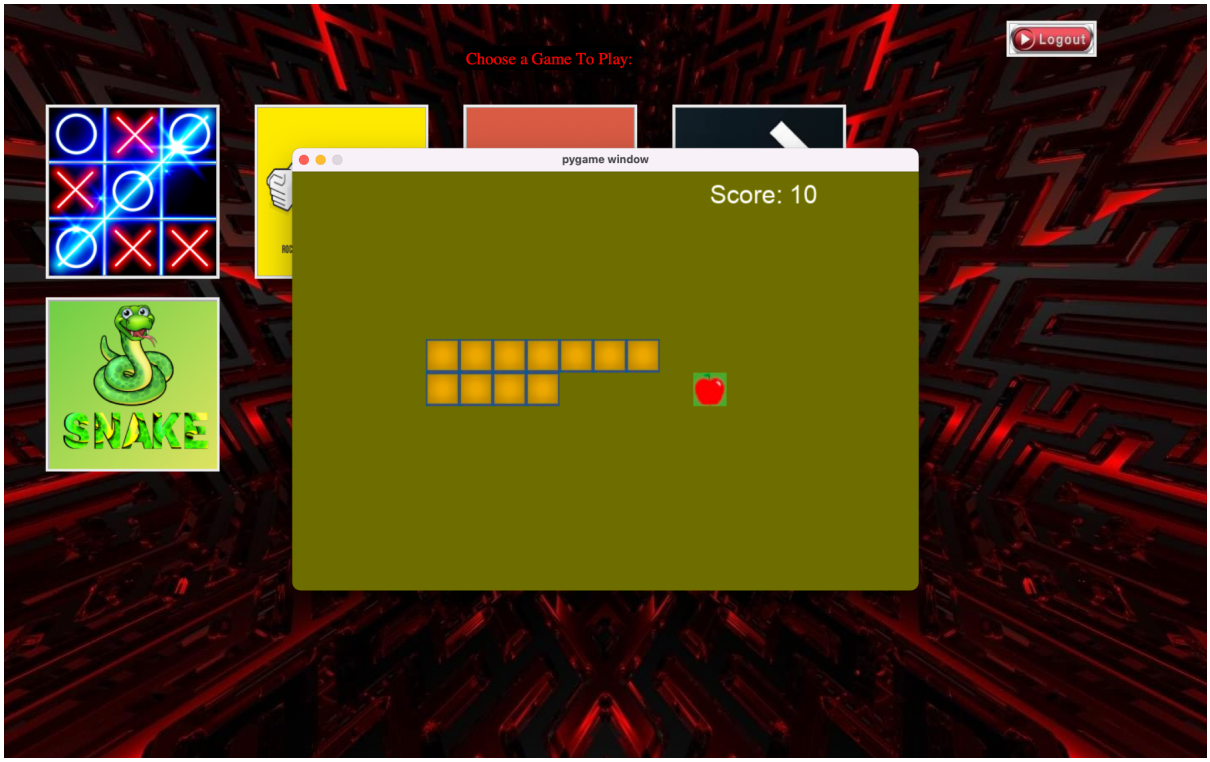
vii) **Ping-Pong?**



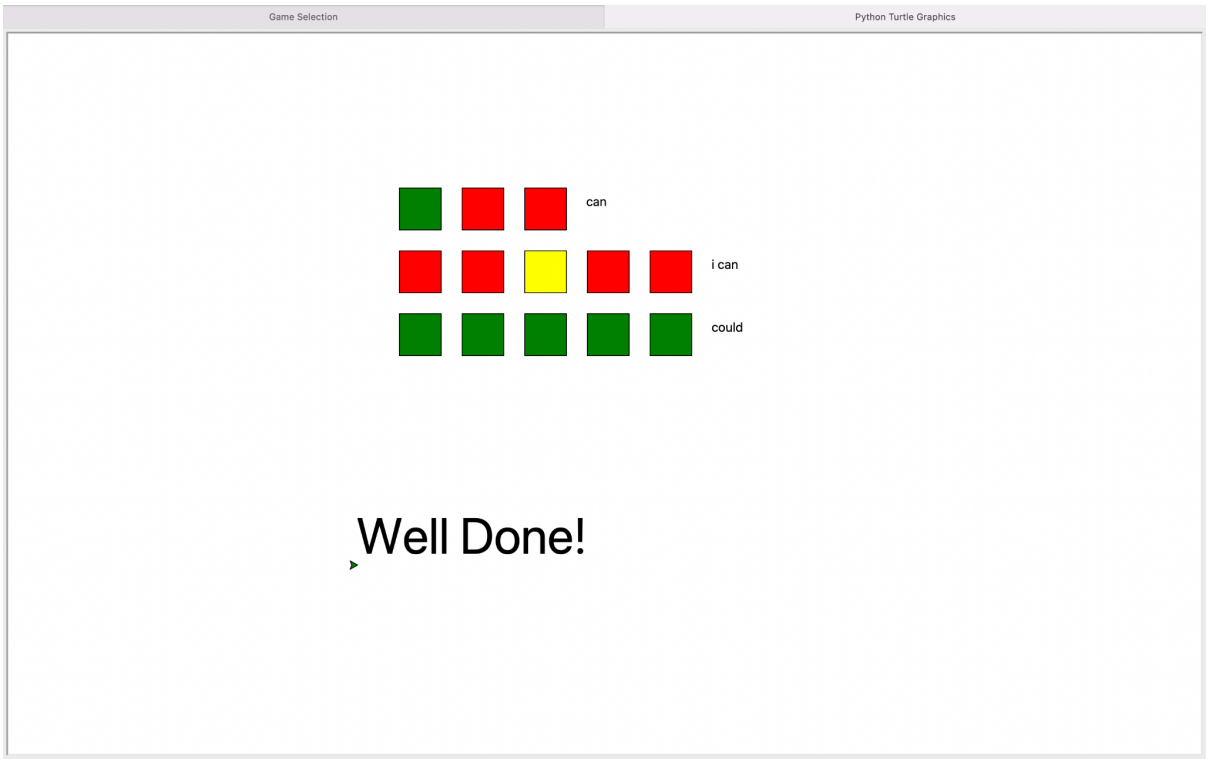
viii) **Rock-Paper-Scissor?**



ix) **Snake and Apple Game?**



x) **Wordle?**



xi) **Enough fun for the day? Well you can logout if you want to.**

6) Future Enchancement:

This project was a real roller-coaster for us. We had ups and downs but we did learn a lot of things in the code developer world. This project is not perfect and will never be, no code or system is perfect, but we will always strive towards it.

Here are a couple of things we have kept as our future project enhancement goals in our mind:

1) Add a User scoreboard, Game Avatars and Highest scores in individual games.

2) Shift the Information storage from Text files to high-end SQL to help us manage data more efficiently.