# COMPUTER SCIENCE PROJECT ON

## FIFA PRO

| | |
|---|---|
| Name | Arnav Gupta |
| Class | XII-B |
| Reg. No. | |
| Academic Year | 2022-2023 |

Prepared as partial fulfilment of requirement in the subject as per guidelines issued by Central Board of Secondary Education, New Delhi

THE DEENS
ACADEMY

# CERTIFICATE

This is to certify that Mr./Ms. Arnav Gupta of Grade-XII, with Roll Number ____________________ of The Deens Academy School, Bangalore has completed his/her project in the subject of Computer Science for the Grade XII practical examination of the Central Board of Secondary Education for the academic year 2022-2023 under my supervision.

He/ She has taken great care and shown utmost sincerity in the completion of the project.

I further certify that this project is up to my expectation and is as per C.B.S.E requirements.

Internal Examiner

Shanthi Menon
Principal

External Examiner

## ACKNOWLEDGEMENT

*"It is easy to acknowledge, but almost impossible to realize for long, that we are mirrors whose brightness, if we are bright, is wholly derived from the sun that shines upon us." -*

*CS Lewis*

I would like to thank my principal, Mrs. Shanthi Menon, for providing the resources for the project.

I would also like to thank our teacher, Mrs. Rajeswari, for her constant guidance and motivation throughout the making of the project, without which the project would never have come to fruition.

I would also like to extend my gratitude to my parents who have given valuable inputs and resources for the completion of this project.

Finally, I would like to express my deep appreciation towards my teammate who showed great dedication towards the project and worked tirelessly along with me to finish it.

Any omission in this brief acknowledgement does not imply a lack of gratitude.

# INDEX

# INTRODUCTION

Python was initially designed by Guido Van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis code reliability and its syntax allows developers to express concepts in fewer lines of code.

Python has a simple and easy to learn syntax. Its syntax also allows the code to be brief and easy to understand. The fact that it works in a cross platform format allowed me and my teammate to collaborate with ease. Online collaborative spaces like Google Collab also helped a great deal in the same regard. The fact that it is an interpreted language made it easier to debug and code.

FIFAPRO is an online auction system that uses socket programming to provide the user with a way to auction soccer players and play with the same players against the opponent

## THEORY

This project is a multiplayer auction system where one can auction for a player against one’s opponent. It also allows the user to play with the newly created team.

The program can be executed as a stand alone or in a multi-user environment. Where the user gets to choose the window as a server or client.

The auction for team members begins between the 2 clients and they are added to the base team.

Once the auction system is over and the team has been created (6 players each), football card game starts. The game is in a similar format to an online card game. All the damages, moves and stamina are decided based on the rating of the player. The game continues until a team member has lost all his stamina whereby the player is asked to select another player from the remaining members. When all team members of the opponent team have been defeated by the player he earns one goal and the game ends.

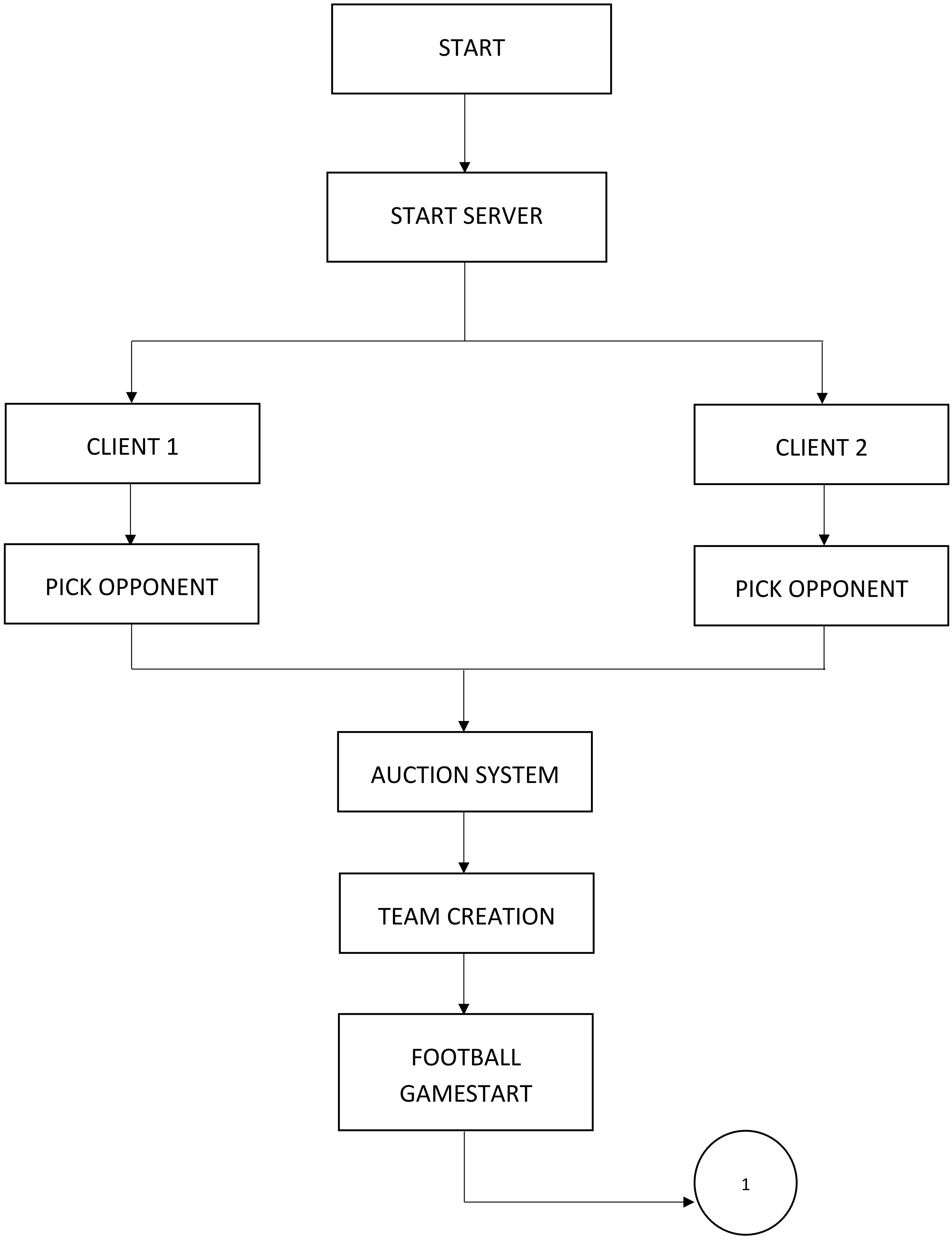
## SYSTEM REQUIREMENTS

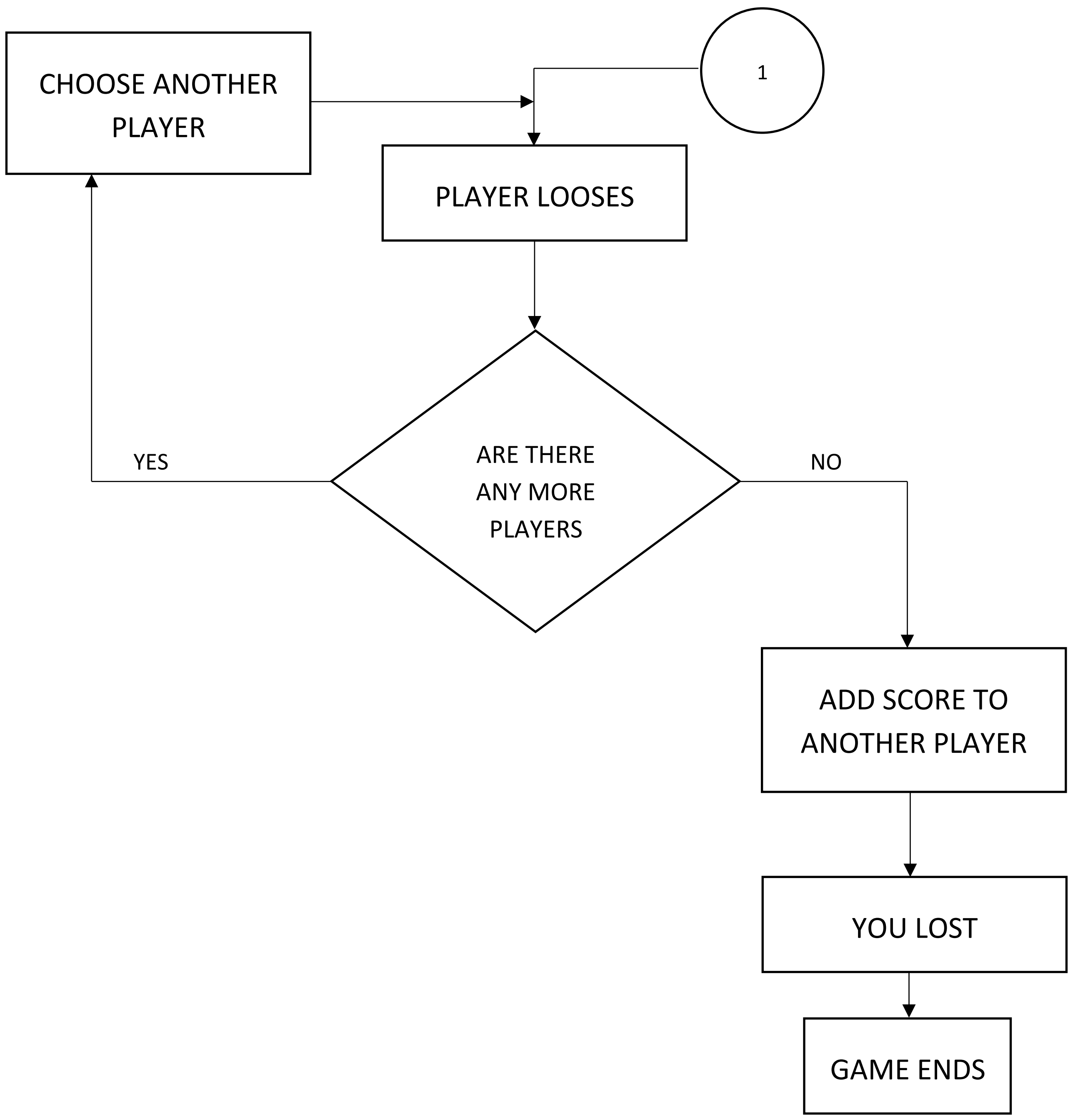
| | |
|---|---|
| **RAM** | 8 GB |
| **Microprocessor** | Intel® Core™ i5-6200U CPU @ 2.30GHz 2.40GHz |
| **System Type** | 64-bit operating system, x64-based processor |
| **Operating System** | Windows 10 Home Single Language |

## SYSTEM DESIGN

1. Tkinter: for GUI
2. PIL: for importing pictures
3. Socket: for making inter system connections
4. Threading: to run other processes in the background
5. Mysql.connector: for creating and accessing mysql databases
6. Random: Generates random numbers and randomises lists
7. Time: Stalls code for specific amount of time
8. distributeMoney: Runs the auction program

# FLOW OF CONTROL




CHOOSE ANOTHER PLAYER
1
PLAYER LOOSES
ARE THERE ANY MORE PLAYERS
YES
NO
ADD SCORE TO ANOTHER PLAYER
YOU LOST
GAME ENDS

# SOURCE CODE

**INTEGRATED GUI**

```
from tkinter import *
import mysql.connector as mysql
import dataloading as dtl
import auctionsys as actsys
from networking import *
import tkinter.messagebox
import threading as thr
import time
from players import team_creation
from game import game_working
from PIL import Image,ImageTk

passw = None
bg= "black"
fg = "white"
def destruct(ele):
    for i in ele:
        print(i)
        i.destroy()

otherPlayer = None
root = Tk()
class start:
    def popup(self):
        tkinter.messagebox.showinfo('info','''HOW THE GAME WORKS:

This game is a two player game
This game function similar to a card based game

The game starts with an auction system where both players bid for two players given in the auction window. They can either raise bid or backout. Once backed out The opponent player receives the player.

After players have bided. A randomiser alocats the rest of the team to each player with 6 in a team.
Then the game starts.

each team member has a set stamina(health) and 3 moves.

each player chooses one team member.  For each turn a player selects a move and does damage to the other player's team member's health. when the team member has 0 stamina the player has to pick another team member from their team. When all the team members team has been defeated the player loses and opponent player wins and vice versa

    ''')
    def __init__(self,root):
        self.root=root
        d = [1200,750]
        self.root.geometry(f"{d[0]}x{d[1]}")
        self.root.resizable(False,False)
        img = Image.open("start.png")
        resize = img.resize((d[0],d[1]), Image.ANTIALIAS)
        image= ImageTk.PhotoImage(resize)
        self.img = Label(root, image=image)

        self.img.place(x=0,y=0)

        b = Button(root,command = self.popup, text= 'i',padx = 5,pady = 5,borderwidth =
0,width = 2)

        b.place(x=650*d[0]/700,y=d[1]*30/400)


        self.ele = [b,self.img]


        b1 = Button(root,command = self.next, text= 'proceed',padx = 5,pady = 5,borderwidth =
0)

        b1.place(x=d[0]*300/700,y=d[1]*350/400)

        self.ele.append(b1)

        self.l = Label(self.root,text = "FIFA PRO",font = "Helvetica",padx = 10,pady =
10,foreground = "white",background = "black")

        self.l.place(x=d[0]*300/700,y=d[1]*10/400)

        self.ele.append(self.l)

        root.mainloop()

    def next(self):

        destruct(self.ele)

        serverOrClient()

class serverOrClient:

    def __init__(self,master = root):

         self.ele = []

         self.master=master

         self.master.geometry("700x400")

         self.master.title("FIFA")

         self.master.resizable(False,False)
         self.clientor = Label(self.master,text = "DO YOU WANT TO SET UP THE DEVICE AS A
CLIENT OR SERVER")

         self.clientor.place(x = 150,y = 70)

         self.ele.append(self.clientor)


         self.client =
Button(self.master,text="CLIENT",padx=80,pady=30,borderwidth=0,command =lambda:\

                    self.setAsClient())


         self.server =
Button(self.master,text="SERVER",padx=78,pady=30,borderwidth=0,command =lambda:\

                    self.setAsServer())


         self.ele.append(self.client)

         self.ele.append(self.server)

         self.client.place(x = 100,y = 200)

         self.server.place(x = 358,y=200)



         root.mainloop()

    def setAsServer(self):

        global comp

        comp = Server()

        self.next_window(False)

    def setAsClient(self):

        global comp

        comp = Client()

        self.next_window(True)
    def next_window(self,isClient):
        destruct(self.ele)
        if isClient:enterUser()
        else:server_screen(root)


#comp = None
class enterUser:
    def __init__(self,root=root):
        self.ele = []
        self.entryfg = "red"


        self.root = root
        self.root.resizable(False,False)
        self.root.geometry(f"600x150+5+5")
        self.root.configure(bg= bg)
        self.root.title("a")




        self.ele.append(Label(self.root,text = "Enter your username(8+character, dashes and
colon are not allowed):",font = "Helvetica",padx = 10,pady = 10,foreground = fg,background
= bg))
        self.ele[-1].place(x = 0, y=  0)


        self.enterbox = Entry(self.root,width = 60,background = "white", foreground =
self.entryfg)
        self.enterbox.place(height = 30,x = 10,y = 50)
        self.enterbox.bind("<Key>",self.checkName)
        self.ele.append(self.enterbox)
        self.l =  Label(self.root,text = "",font = "Helvetica",padx = 10,pady = 10,foreground = 
fg,background = bg)

        self.l.place(x = 10, y=  100)

        self.ele.append(self.l)


        self.root.mainloop()

    def subm(self):

        c = self.enterbox.get()

        if '-' in c or ":" in c or len(c)<8:

            self.l.config(text = "Please enter a 8+ character username without dashes or colon")

        else:

            comp.user = c

            self.enterbox.delete(0,END)

            destruct(self.ele)

            try:

                f= open("server_details.txt","r")

                global passw

                passw = f.read()

                connectServer()

            except:

                mysql_password(root)


    def checkName(self,key):

            if ord(key.char) == 13:

                self.subm()

                return

            c = self.enterbox.get()
        if "-" not in c and ":" not in c and len(c)>=8:
            self.entryfg = "green"
        else:
            self.entryfg = "red"
        self.enterbox.config(foreground = self.entryfg)

class mysql_password:
    def popup(self):
        tkinter.messagebox.showinfo('info','''Refer to your system's card for the mysql
password''')
    def __init__(self,master):
     global b
     self.master=master
     self.b = Button(root,command = self.popup, text= 'i',padx = 5,pady = 5,borderwidth =
0,width = 2)
     self.b.place(x = 650,y = 10)
     self.master.geometry("700x400")
     self.master.title("FIFA")
     b = StringVar()
     self.master.configure(bg = "black")

     self.password = Label(self.master,bg = "black",fg = "white",text = "Please enter your
system's mysql password and press enter",font = ("Calibri",17))
     self.password.pack()

     self.entry = Entry(self.master,text=b,show = "*",font = ("Calibri",17))
     self.entry.pack()
     self.entry.bind("<Return>",self.checkpss)
     self.ele = [self.password,self.entry]
```

```
  def checkpss(self,k):
   if ord(k.char)!=13:
       return
   pss = str(b.get())


   try:
       test = mysql.connect(host = "localhost",user = "root", passwd = pss)
       destruct(self.ele)
       with open("server_details.txt","w") as user:
           mysql_pss = pss
           user.write(pss)
       global passw
       passw = pss
       connectServer()
   except mysql.errors.ProgrammingError:
       pass


class server_screen:
  def __init__(self,root=root):
      self.r = root
      self.r.geometry("400x300")
      self.r.configure(bg = "black")
      self.ele = []
      self.label = Label(self.r,text = f"Your ip address is : {convertih(myip)}",font =
("Calibri",20),bg = "black",fg ="white")
      self.label.pack()
      self.ele.append(self.label)
      self.b =Button(self.r,text="close this server",command = self.close,bg = "black",fg =
                  "white")
      self.b.pack()
      self.ele.append(self.b)
  def close(self):
      global comp
      del comp
      self.r.destroy()
      del self
class connectServer:
  def __init__(self,root = root):
      self.t = thr.Thread(target = self.load)
      self.t.start()
      self.root = root
      self.root.configure(bg = "black")
      self.root.geometry("700x400")
      self.l = Label(self.root,text = "Enter the server ip you want to join and press enter",font = 
("Helvetica",13),padx = 10,pady = 10,fg = "white",bg ="black" )
      self.ele = [self.l]
      self.l.pack()
      self.enterbox = Entry(self.root,width = 60,background = "white")
      self.enterbox.pack()
      self.enterbox.bind("<Return>",self.connServer)
      self.ele.append(self.enterbox)
  def load(self):
      dtl.maketable("players_fifa22",passw)
      dtl.loadData("players_fifa22",passw)
  def connServer(self,key):
      c = self.enterbox.get()
      print(c)

      if len(c)!=8:

          self.l1 = Label(self.root,text = "The server you entered may not be running",font =
("Helvetica",13),padx = 10,pady = 10,fg = "white",bg ="black" )

      x = comp.setip(self.enterbox.get())

      if x:

          self.t.join()

          destruct(self.ele)

          join(root)

      else:

          self.l1 = Label(self.root,text = "The server you entered may not be running",font =
("Helvetica",13),padx = 10,pady = 10,fg = "white",bg ="black" )

class join():

  def __init__(self,root):

      self.root = root

      self.root.configure(bg = "black")

      self.root.geometry("700x400")

      self.ele = []

      self.l = None

      self.cont = True

      t = thr.Thread(target=self.updateList)

      t.start()

      t2 = thr.Thread(target=self.transition)

      t2.start()

      self.lab = Label(self.root,text = "Pick who you want to play with:",font =
("Helvetica",13),padx = 10,pady = 10,fg = "white",bg ="black" )

      self.lab.pack()

      time.sleep(.1)
      self.lab1 = Label(root,text = f"your username is: {comp.user}",bg = 'black',font=("Arial",
13),fg = "white")

      self.lab1.pack()


  def updateList(self):

      while True:

          if not self.cont:

              break

          if self.l !=comp.servconn:

              destruct(self.ele)

              print(2)

              self.l = comp.servconn

              n = 0

              for i in range(len(self.l)):

                  if self.l[i]!= comp.user:

                      n+=1

                      self.ele.append(Button(self.root,padx = 200,pady =
3,text=self.l[i],borderwidth=0))

                      x= self.ele[-1]

                      self.ele[-1].configure(command =lambda a=i,b=x: self.joinconn(self.l[a],b))

                      self.ele[-1].place(relx=0.5,rely=n*0.1+0.17,anchor='center')

  def joinconn(self,user2,b):

      print("J")

      b.configure(bg= "blue")

      comp.write(f"::cnct-{comp.user}:{user2}")

  def transition(self):

      while True:

          if comp.paired:
```

```
            self.cont = False

            print(self.ele)

            destruct(self.ele+[self.lab,self.lab1])

            distributeMoney(root)

            break


class message:

    def __init__(self,root=root):

        self.root = root

        self.root.configure(bg = "black")

        self.root.geometry("700x400")

        self.enterbox = Entry(self.root,width = 60,background = "white")

        self.enterbox.pack()

        self.enterbox.bind("<Return>",self.onEnter)

    def onEnter(self,k):

        comp.write(f"{comp.opp}-{self.enterbox.get()}")

moneyToTeam = None

money = 1000

class distributeMoney:

    def popup(self):

        tkinter.messagebox.showinfo('info','''Team investment:
```

Out of 50 randomly selected players, the base team of the user is made by prioritizing higher value players and the strength of this priority is determined by how much money the user decides to put into forminghis baseteam.

Auction players:

These are high ranked players chosen so that users can battle for them witth the remainder of their money.

Auction system:

```
    Click on raise bid by 100 if you want to increase the bid for a player.

    The player's name should turn green if you have the highest bid for that player and red if your opponent has the highest bid.

    If you want to back out of the bid for a player, you can only do so if your opponent has the highest bid.

    You cannot set a bid for a player if the total bid amount crosses the money you have left.



    ''')

    def __init__(self,root):

        global comp

        comp.eventhand = self.evnt

        self.bid = [0,0]

        self.bidDone = [False,False]

        self.root=root

        self.b = Button(root,command = self.popup, text= 'i',padx = 5,pady = 5,borderwidth = 0,width = 2)

        self.b.place(x=650,y=30)

        self.root.configure(bg = "black")

        self.root.geometry("700x400")

        self.ele = []

        if comp.user>comp.opp:

            self.p2 = actsys.pickAuctionPlayer(passw)   #name,ovr,pos

            self.p1 = actsys.pickAuctionPlayer(passw)


            comp.write(f"{comp.opp}-auctplrs:{self.p1},{self.p2}")

        time.sleep(.2)

        self.label1 = Label(root,text = f"bid:{self.bid[0]}",bg = 'black',font=("Arial", 20),fg = "white")
        self.label1_ = Label(root,text = f"bid:{self.bid[1]}",bg = 'black',font=("Arial", 20),fg =
"white")

        self.ele.append(Label(self.root,text = "how much of your money do you want to invest
in auction?(out of 1000)",font = ("Helvetica",13),padx = 10,pady = 10,fg = "white",bg
="black" ))

        self.ele[-1].pack()

        self.ele.append(Entry(self.root,width = 60,background = "white"))

        self.ele[-1].pack()

        self.enterb = self.ele[-1]

        self.enterb.bind("<Key>",self.sub)

        self.l = Label(self.root,font = ("Helvetica",13),padx = 10,pady = 10,fg = "white",bg
="black" )

        self.l.pack()

        self.ele.append(self.l)

    def sub(self,key):

        if ord(key.char)!=13:

            return

        c = self.enterb.get()

        if c.isdigit() and 0<=int(c)<=1000:

            global moneyToTeam,money

            moneyToTeam = 1000-int(c)

            money= int(c)

            destruct(self.ele)

            self._init__(root)

        else:

            self.l.config(text = "Please enter a valid integer between 0 and 1000")

    def _init__(self,root):

        global comp

        canvas = Canvas(root, width = 660, height = 400, bg = 'black',relief = 'sunken')
        canvas.place

        self.root = root

        self.canvas = canvas

        self.moneyLabel = Label(root,text = f"initial money:{money}",bg = 'black',font=("Arial",
20),fg = "white")

        root.geometry("700x400")

        root.title("FIFA")

        root.resizable(False,False)

        root.config(bg = 'black')

        self.label2 = Label(root,text = "name:"+self.p1[0],bg = 'black',font=("Arial",
20),fg="white")

        self.label3 = Label(root,text = "rating:"+str(self.p1[1]),bg = 'black',font=("Arial", 20),fg =
"white")


        self.label1.place(x = 30,y = 90)

        self.label2.place(x = 30,y = 150)

        self.label3.place(x = 30,y = 210)


        self.label2_ = Label(root,text = "name:"+self.p2[0],bg = 'black',font=("Arial", 20),fg =
"white")

        self.label3_ = Label(root,text = "rating:"+str(self.p2[1]),bg = 'black',font=("Arial", 20),fg =
"white")


        self.label1_.place(x = 390,y = 90)

        self.label2_.place(x = 390,y = 150)

        self.label3_.place(x = 390,y = 210)


        b1 = Button(root,command = lambda:self.raisebid(1), text= 'RAISE BID BY 100',padx =
50,pady = 12,borderwidth = 0,width = 2)
```

```
        b2 = Button(root,command = lambda:self.back(1), text= 'BACKOUT',padx = 50,pady = 
12,borderwidth = 0,width = 2)

        b3 = Button(root,command = lambda:self.raisebid(2), text= 'RAISE BID BY 100',padx = 
50,pady = 12,borderwidth = 0,width = 2)

        b4 = Button(root,command = lambda:self.back(2), text= 'BACKKOUT',padx = 50,pady = 
12,borderwidth = 0,width = 2)

        self.moneyLabel.place(x=250,y=5)

        b1.place(x = 20, y = 320)

        b2.place(x = 175, y = 320)

        b3.place(x = 377, y = 320)

        b4.place(x = 533, y = 320)

        self.ele = 
[self.canvas,self.label1,self.moneyLabel,self.label2,self.label3,self.label1_,self.label2_,self.lab
el3_,b1,b2,b3,b4]

    def raisebid(self, auctplr):

        if money<100+self.bid[0]*(auctplr==1 or 
self.label1.cget("fg")=="green")+self.bid[1]*(auctplr==2 or self.label1_.cget("fg")=="green"):

            return

        comp.write(f"{comp.opp}-raisebid:{auctplr}")

        print(f"{comp.opp}-raisebid:{auctplr}")

        self.evnt([f"raisebid:{auctplr}"],True)

    def back(self,plr):

        if (self.label1.cget("fg")!="red" and plr == 1) or (self.label1_.cget("fg")!="red" and plr == 
2):

            return

        print(self.label1.cget("fg"))

        comp.write(comp.opp+"-back:"+str(plr))

        self.evnt(("back:"+str(plr)).split("-"),True)

    def evnt(self,msg,msgsentbyself = False):

        print(msg)
```

```
msg = msg[0].split(":")
if msg[0] == "raisebid":


    print((self.bid,msg[1]))
    if (not self.bidDone[0]) and msg[1]=="1":
        self.bid[0]+=100
        self.label1.configure(text = "bid:"+str(self.bid[0]))
        if msgsentbyself:
            self.label1.configure(fg = "green")
        else:
            self.label1.configure(fg = "red")
    elif (not self.bidDone[1]) and msg[1]=="2":
        self.bid[1]+=100
        self.label1_.configure(text = "bid:"+str(self.bid[1]))
        if msgsentbyself:
            self.label1_.configure(fg = "green")
        else:
            self.label1_.configure(fg = "red")


elif msg[0]=="back":
    print(5)
    if msg[1]=="1":
        self.label1.configure(text = "bid:"+str(self.bid[0])+"(confirmed)")
        self.bidDone[0] = True
    elif msg[1]=="2":
        self.label1_.configure(text = "bid:"+str(self.bid[1])+"(confirmed)")
        self.bidDone[1] = True
    if self.bidDone[0] and self.bidDone[1]:
        print(moneyToTeam/2000+1)
        team = actsys.team_init(moneyToTeam/500+1,passw)
        if self.label1.cget("fg")=="green":
            if self.p1[2] in ["LB","RB","CB","CDM","CM","GK"]:
                mode = 0
            else:
                mode = 1
            lowest = [None,0]
            for i in range(3):
                if lowest[1]<team[mode][i][1]:
                    lowest = [i,team[mode][i][1]]
            team[mode][lowest[0]] = self.p1
        if self.label1_.cget("fg")=="green":
            if self.p2[2] in ["LB","RB","CB","CDM","CM","GK"]:
                mode = 0
            else:
                mode = 1
            lowest = [None,0]
            for i in range(3):
                if lowest[1]<team[mode][i][1]:
                    lowest = [i,team[mode][i][1]]
            team[mode][lowest[0]] = self.p2
        print(team)
        destruct(self.ele)
        gamework(root,team)
elif msg[0]=="auctplrs":
```

```
            self.p1,self.p2 = eval(msg[1])


class gamework:


    def __init__(self,master,team):
        self.team_final = []
        comp.eventhand = self.event_handler


        team[0].extend(team[1])
        team_new = team[0]


        print()
        print("see:",team_new)
        print()


        print(comp.user)
        team_new.insert(0,comp.user)
        self.team_final.append(team_new)
        self.send(f"{team_new}")


    def event_handler(self, event):
        value = eval(event[0])
        opp_team = value
        self.team_final.append(opp_team)
        print("final team:",self.team_final)
        self.player_moves()
        game_start(root)
    def player_moves(self):
        f = team_creation(root,comp,self.team_final)

    def send(self,msg):
        print("send",msg)
        comp.write(f"{comp.opp}-{msg}")

class game_start:
    def __init__(self,root):
        g = game_working(root,comp)

start(root)
```

**GAME CODE**

```
from tkinter import *
import pickle
from PIL import Image,ImageTk
import time



class game_working:
    def __init__(self,root,comp):
        self.comp = comp
        self.comp.eventhand = self.event_handler

        #screen
        self.root = root
        self.root.geometry("1050x750")
        self.root.title("FIFAPRO")

        self.root.resizable(False,False)

        self.urteam()

        self.player = None
        self.playing = []
        self.score = 0
        self.selfscore = 0
        self.count_turn = 0
        self.remaining = []



        self.turn = 0

        self.opp_player = None

        self.opp_player_test = None


        self.canvas = Canvas(self.root, width = 1050, height = 650, bg = 'black',relief = 'sunken')

        self.canvas.pack()


        self.canvas.create_rectangle(771,180,1018,310,fill ='black', outline = 'white')


        self.canvas.create_text(1000, 200, anchor='e',text='OPPONENT SCORE', fill="white",
font=('Arial 10 '))

        self.canvas.create_text(781, 200, anchor='w',text='YOUR SCORE', fill="white",
font=('Arial 10 '))


        self.canvas.create_text(826, 250, anchor='w',text=self.selfscore, fill="white", font=('Arial
15 '))

        self.canvas.create_text(900, 250,text=':', fill="white", font=('Arial 15 '))

        self.canvas.create_text(960, 250, anchor='e',text='0', fill="white", font=('Arial 15 '))


        self.canvas.create_rectangle(471,180,741,310,fill ='black', outline = 'white')




        self.choose_player()
        #football ground

        img = Image.open("football_field.png")

        resize = img.resize((480,385), Image.ANTIALIAS)

        rotated = resize.rotate(90,expand = True)

        self.new_image= ImageTk.PhotoImage(rotated)

        self.canvas.create_image(224,250,image=self.new_image)



        self.canvas.create_rectangle(20,500,426,635,fill ='blue',outline = 'blue')


        self.canvas.create_rectangle(471,330,741,635,fill ='black', outline = 'white')


        #health bar


        self.x1 = 416

        self.x_ = 416


        self.canvas.create_rectangle(29,451,250,470,fill ='black',outline = 'black')

        self.canvas.create_rectangle(29,42,380,65,fill ='black',outline = 'black')

        self.canvas.create_text(30, 460, anchor='w',text='PLAYER HEALTH BAR', fill="white",
font=('Arial 12 '))

        player_bar = self.canvas.create_rectangle(30,480,416,505,fill ='green',outline = 'white')

        self.canvas.create_text(30, 50, anchor='w',text='OPPONENT HEALTH BAR AND PLAYER',
fill="white", font=('Arial 12 '))

        opponent_bar = self.canvas.create_rectangle(30,10,416,35,fill ='red',outline = 'white')
```

```
    with open("team.dat","rb") as file:
        l =[]

        t = pickle.load(file)
        for i in range(len(t)):
            if t[i][0] == self.comp.user:
                for j in range(1,len(t[i])):
                    l.append(t[i][j][0].upper())


        self.f_players = Listbox(self.canvas,width = 27, height = 13,bg = 'black',fg = 
'white',font = 'Arial',activestyle = 'none',bd = 2,relief = 'sunken')

        for item in l:
            self.f_players.insert('end',item)

        def select():
            selected_name = self.f_players.get(ANCHOR)
            if selected_name != '':
                self.card(selected_name)



        select_option1 = Button(self.root, text= 'select' ,bg = 'black',fg = 'white',padx = 
76,pady = 15,command = lambda: select(),borderwidth = 0,width = 4)
        select_option1.place(x =803,y =590)
```

```
            self.f_players.place(x = 771, y = 330)


    def urteam(self):
        self.pop = Toplevel(self.root)
        self.pop.geometry("910x670")
        self.pop.config(bg = "black")
        self.pop.resizable(False,False)
        self.pop.title("YOUR TEAM")

        self.canva = Canvas(self.pop, width = 910, height = 670, bg = 'black',relief = 'sunken')
        self.canva.pack()

        length = 30
        width = 30
        count = 0

        with open("team.dat","rb") as file:

            t = pickle.load(file)
            for i in range(len(t)):
                if t[i][0] == self.comp.user:
                    for j in range(1,len(t[i])):
                        count +=1

                        n = t[i][j][0].upper()
                        st = str(t[i][j][5])
                        m1 = t[i][j][2][0].upper()

                        m2 = t[i][j][3][0].upper()

                        m3 = t[i][j][4][0].upper()


                        d1 = str(t[i][j][2][1])

                        d2 = str(t[i][j][3][1])

                        d3 = str(t[i][j][4][1])


                        if count <= 3:


                            self.canva.create_rectangle(length-10,20,length+260,325,fill ='black',
        outline = 'white')




                            name = self.canva.create_text(length, 30, anchor='w',text=n, fill="white",
        font=('Arial 12 '))

                            stamina = self.canva.create_text(length+240, 30, anchor='e', text=st,
        fill="white", font=('Arial 13 '))


                            move1 = self.canva.create_text(length, 130, anchor='w', text=m1,
        fill="white", font=('Arial 12 '))

                            damage1 = self.canva.create_text(length+240, 130, anchor='e', text=d1,
        fill="white", font=('Arial 13 '))


                            move2 = self.canva.create_text(length, 160, anchor='w', text=m2,
        fill="white", font=('Arial 12 '))

                            damage2 = self.canva.create_text(length+240, 160, anchor='e',text=d2,
        fill="white", font=('Arial 13 '))
```

```
        move3 = self.canva.create_text(length, 190, anchor='w', text=m3,
fill="white", font=('Arial 12 '))

        damage3 = self.canva.create_text(length+240, 190, anchor='e', text=d3,
fill="white", font=('Arial 13 '))


        if count == 3:

          length = 30


      elif count > 3:

        self.canva.create_rectangle(length-10,345,length+260,650,fill ='black',
outline = 'white')



        name = self.canva.create_text(length, 30+345, anchor='w',text=n,
fill="white", font=('Arial 12 '))

        stamina = self.canva.create_text(length+240, 30+345, anchor='e', text=st,
fill="white", font=('Arial 13 '))


        move1 = self.canva.create_text(length, 130+345, anchor='w', text=m1,
fill="white", font=('Arial 12 '))

        damage1 = self.canva.create_text(length+240, 130+345, anchor='e',
text=d1, fill="white", font=('Arial 13 '))


        move2 = self.canva.create_text(length, 160+345, anchor='w', text=m2,
fill="white", font=('Arial 12 '))

        damage2 = self.canva.create_text(length+240, 160+345,
anchor='e',text=d2, fill="white", font=('Arial 13 '))


        move3 = self.canva.create_text(length, 190+345, anchor='w', text=m3,
fill="white", font=('Arial 12 '))

        damage3 = self.canva.create_text(length+240, 190+345, anchor='e',
text=d3, fill="white", font=('Arial 13 '))
```

```
                    length += 300



        time.sleep(5)
        self.pop.destroy()



    def move_buttons(self,player):
        with open("team.dat","rb") as file:

                t = pickle.load(file)
                for i in range(len(t)):
                    if t[i][0] == self.comp.user:
                        for j in range(1,len(t[i])):
                            if player.lower() ==  t[i][j][0].lower():

                                self.opp_player = t[i][j][0].upper()

                                self.card(t[i][j][0])
                                self.playing.append(t[i][j][0].upper())
                                damage1 = t[i][j][2][1]
                                damage2 = t[i][j][3][1]
                                damage3 = t[i][j][4][1]
                                self.b1 = Button(self.root, text=t[i][j][2][0],padx = 70,pady =
15,borderwidth = 0,command = lambda: self.health_bar(damage1),width = 6)
                                self.b2 = Button(self.root, text=t[i][j][3][0],padx = 70,pady =
15,borderwidth = 0,command = lambda: self.health_bar(damage2),width = 6)
                                self.b3 = Button(self.root, text=t[i][j][4][0],padx = 70,pady =
15,borderwidth = 0,command = lambda: self.health_bar(damage3),width =6)


                                self.b1.place(x = 30, y = 510)
                                self.b2.place(x = 230, y = 510)
                                self.b3.place(x = 30, y = 575)



                                self.send(f'starting_player:{self.opp_player}')


                                self.canvas.create_rectangle(30,480,416,505,fill ='green',outline = 'white')


        self.pop.destroy()


    def card(self,x):
        self.canvas.create_rectangle(481,340,731,625,fill ='black',outline = 'black')


        with open("team.dat","rb") as file:
                t = pickle.load(file)
                for i in range(len(t)):
                    if t[i][0] == self.comp.user:
                        for j in range(1,len(t[i])):
                                if t[i][j][0].lower() == x.lower():
                                n = t[i][j][0].upper()
                                st = str(t[i][j][5])
                                m1 = t[i][j][2][0].upper()

                                m2 = t[i][j][3][0].upper()

                                m3 = t[i][j][4][0].upper()


                                d1 = str(t[i][j][2][1])

                                d2 = str(t[i][j][3][1])

                                d3 = str(t[i][j][4][1])




                                name = self.canvas.create_text(486, 350, anchor='w',text=n, fill="white",
font=('Arial 12 '))

                                stamina = self.canvas.create_text(726, 350, anchor='e', text=st,
fill="white", font=('Arial 13 '))


                                move1 = self.canvas.create_text(486, 450, anchor='w', text=m1,
fill="white", font=('Arial 12 '))

                                damage1 = self.canvas.create_text(726, 450, anchor='e', text=d1,
fill="white", font=('Arial 13 '))


                                move2 = self.canvas.create_text(486, 480, anchor='w', text=m2,
fill="white", font=('Arial 12 '))

                                damage2 = self.canvas.create_text(726, 480, anchor='e',text=d2,
fill="white", font=('Arial 13 '))


                                move3 = self.canvas.create_text(486, 510, anchor='w', text=m3,
fill="white", font=('Arial 12 '))

                                damage3 = self.canvas.create_text(726, 510, anchor='e', text=d3,
fill="white", font=('Arial 13 '))


    def event_handler(self, event):
```

```
keyword,value = event[0].split(":")

if keyword == 'damage':

    x1 = value
    x1 = float(x1)

    self.turn=0
    self.endturn()

    if x1 <= 30:
        self.canvas.create_rectangle(30,480,416,505,fill ='black',outline = 'white')
        with open("team.dat","rb") as file:
            l =[]

            t = pickle.load(file)
            for i in range(len(t)):
                if t[i][0] == self.comp.user:
                    for j in range(1,len(t[i])):
                        l.append(t[i][j][0].upper())

        len_l = len(l)
        len_playing = len(self.playing)

        if len_l == len_playing:
            self.score += 1
```

```
                self.send(f'your_score:{self.score}')

                self.canvas.create_rectangle(940,240,1000,270,fill = 'black',outline = 'black')

                self.canvas.create_text(960, 250, anchor='e',text=f'{self.score}', fill="white",
font=('Arial 15 '))


                self.lost()





              else:

                self.change_player()


            else:

              self.canvas.create_rectangle(x1,480.5,415,504,fill = 'black',outline = 'black')


          elif keyword == 'opp_name_change':


            self.opp_player = value

            self.opp_player_test = self.opp_player

            self.canvas.create_rectangle(30,10,416,35,fill ='red',outline = 'white')

            self.canvas.create_rectangle(420,5,1000,90,fill ='black',outline = 'black')

            self.canvas.create_text(436, 20, anchor='w',text=self.opp_player, fill="white",
font=('Arial 15 '))


          elif keyword == 'starting_player':

            self.opp_player = value

            self.opp_player_test = self.opp_player
            self.canvas.create_rectangle(30,10,416,35,fill ='red',outline = 'white')

            self.canvas.create_rectangle(420,5,1000,90,fill ='black',outline = 'black')

            opp = self.canvas.create_text(436,20, anchor='w',text=self.opp_player, fill="white", font=('Arial 15 '))


        elif keyword == 'won':

            self.won()


        elif keyword == 'your_score':

            self.selfscore = value

            self.canvas.create_rectangle(820,240,870,270,fill = 'black',outline = 'black')

            self.canvas.create_text(826, 250, anchor='w',text=f'{self.selfscore}', fill="white", font=('Arial 15 '))


    def health_bar(self,damage):

        self.turn = 1

        self.endturn()



        with open("team.dat","rb") as file:

            t = pickle.load(file)

            for i in range(len(t)):

                if t[i][0] != self.comp.user:

                    for j in range(1,len(t[i])):

                        if t[i][j][0].lower() ==self.opp_player_test.lower():

                            opp_total_health = t[i][j][5]

                            print()

                            print("opp player",self.opp_player_test)
                            print("opp damage:",opp_total_health)

                            print("damage sent",damage)

                            damage_amo = (damage/opp_total_health)*386

                            print("damage amount:",damage_amo)

                            print()


            self.x1 -= 30

            self.x1 -= damage_amo


            self.x1 += 30


            if self.x1 < 0:

                self.x1 = 0


            if self.x1 <= 30:

                self.send(f'damage:{self.x1}')


                self.canvas.create_rectangle(30,10,416,35,fill ='black',outline = 'white')

                self.x1 = 416


            else:

                self.send(f'damage:{self.x1}')


                self.canvas.create_rectangle(self.x1,11,415,34,fill ='black',outline = 'black')



    def send(self,msg):
```

```
    print("send",msg)
    self.comp.write(f"{self.comp.opp}-{msg}")

def change_info(self,selected_name):


    with open("team.dat","rb") as file:
        l =[]


        t = pickle.load(file)
        for i in range(len(t)):
            if t[i][0] == self.comp.user:
                for j in range(1,len(t[i])):
                    if t[i][j][0].lower() == selected_name.lower():
                        self.playing.append(selected_name)


                        self.b1.config(text = t[i][j][2][0])
                        self.b2.config(text = t[i][j][3][0])
                        self.b3.config(text = t[i][j][4][0])


                        new_damage1 = t[i][j][2][1]
                        new_damage2 = t[i][j][3][1]
                        new_damage3 = t[i][j][4][1]


                        self.b1.config(command = lambda: self.health_bar(new_damage1))
                        self.b2.config(command = lambda: self.health_bar(new_damage2))
                        self.b3.config(command = lambda: self.health_bar(new_damage3))
                    self.send(f'opp_name_change:{selected_name}')

                    self.canvas.create_rectangle(30,480,416,505,fill ='green',outline = 'white')

                    self.card(selected_name)


        self.f_players.delete(0,END)

        for item in self.remaining:

            self.f_players.insert(END,item)


        self.b1['state'] = NORMAL

        self.b2['state'] = NORMAL

        self.b3['state'] = NORMAL

        self.pop.destroy()


    def change_player(self):

        #global pop

        self.pop = Toplevel(self.root)

        self.pop.geometry("480x460")

        self.pop.config(bg = "grey")

        self.pop.resizable(False,False)

        self.b1['state'] = DISABLED

        self.b2['state'] = DISABLED

        self.b3['state'] = DISABLED


        with open("team.dat","rb") as file:

            l =[]


            t = pickle.load(file)
            for i in range(len(t)):
                if t[i][0] == self.comp.user:
                    for j in range(1,len(t[i])):
                        l.append(t[i][j][0].upper())

            label1 = Label(self.pop,text = "YOUR PLAYER HAS BEEN DEFEATED CHOOSE 
    ANOTHER",bg = 'grey',fg = 'black',font=("Arial", 15))
            label1.place(x = 5,y = 20)

            f = Frame(self.pop,height = 700,width = 700,bg = 'grey')
            f.place(x = 10,y = 50)

            def select_choice():
                selected_name = player_choice.get(ANCHOR)
                if selected_name != '':
                    self.change_info(selected_name)
                    self.f_players.delete(selected_name)



            player_choice = Listbox(f,width = 27, height = 13,fg = 'black',font = 'Arial',activestyle = 
    'none',bd = 2,relief = 'sunken')
            player_choice.place(x = 110, y = 50)


            for i in self.playing:
                if i in l:
                    l.remove(i)
```

```
          self.remaining = []

          for item in l:
             self.remaining.append(item)
             player_choice.insert(END,item)

          select_option1 = Button(f, text= 'select' ,bg = 'black',fg = 'white',padx = 76,pady = 
15,command = lambda: select_choice(),borderwidth = 0,width = 4,relief = 'sunken')
          select_option1.place(x =140,y =330)

    def endturn(self):
       if self.turn == 1:
          self.b1['state'] = DISABLED
          self.b2['state'] = DISABLED
          self.b3['state'] = DISABLED
          self.count_turn +=1

          self.canvas.create_rectangle(473,190,739,300,fill ='black',outline = 'black')
          self.canvas.create_text(506, 250, anchor='w',text="OPPONENTS TURN...", fill="white", 
font=('Arial 15 '))

       else:
          self.b1['state'] = NORMAL
          self.b2['state'] = NORMAL
          self.b3['state'] = NORMAL

          self.canvas.create_rectangle(473,190,739,300,fill ='black',outline = 'black')
```

```
            self.canvas.create_text(536, 250, anchor='w',text="YOUR TURN...", fill="white",
font=('Arial 15 '))



    def choose_player(self):
        self.pop = Toplevel(self.root)
        self.pop.geometry("480x460")
        self.pop.config(bg = "grey")
        self.pop.resizable(False,False)

        with open("team.dat","rb") as file:
            l =[]

            t = pickle.load(file)
            for i in range(len(t)):
                if t[i][0] == self.comp.user:
                    for j in range(1,len(t[i])):
                        l.append(t[i][j][0].upper())



            label1 = Label(self.pop,text = "CHOOSE PLAYER",bg = 'grey',fg = 'black',font=("Arial",
15))
            label1.place(x = 140,y = 20)

            f = Frame(self.pop,height = 700,width = 700,bg = 'grey')
            f.place(x = 10,y = 50)

            def PLAYER():
```

```
            self.player = player_choice.get(ANCHOR)

            if self.player != '':

                self.move_buttons(self.player)



        player_choice = Listbox(f,width = 27, height = 13,fg = 'black',font = 'Arial',activestyle = 
'none',bd = 2,relief = 'sunken')

        player_choice.place(x = 110, y = 50)


        for item in l:

            player_choice.insert(END,item)


        select_option1 = Button(f, text= 'select' ,bg = 'black',fg = 'white',padx = 76,pady = 
15,command = lambda: PLAYER(),borderwidth = 0,width = 4,relief = 'sunken')

        select_option1.place(x =140,y =330)



    def lost(self):

        self.pop = Toplevel(self.root)

        self.pop.geometry("450x400")

        self.pop.config(bg = "grey")

        self.pop.resizable(False,False)


        lab = Label(self.pop,text = "YOU LOST",bg = 'grey',fg = 'black',pady = 200,font=("Arial", 
15))

        lab.pack()


        self.send('won:0')
    def won(self):

        self.pop = Toplevel(self.root)

        self.pop.geometry("450x400")

        self.pop.config(bg = "grey")

        self.pop.resizable(False,False)


        lab = Label(self.pop,text = "YOU  WON",bg = 'grey',fg = 'black',pady = 200,font=("Arial",
15))

        lab.pack()






if __name__ == '__main__':


    root = Tk()

    game = game_working(root)

    root.mainloop()
```

**NETWORKING**

```
import socket as sck

import threading as thr

import pickle


def getIP():

   temp = sck.socket(sck.AF_INET,sck.SOCK_DGRAM)

   temp.connect(("8.8.8.8",9000))

   return temp.getsockname()[0]

import tkinter as tk


myip = getIP()



def read(x,comp):

 try:

   while True:

       msg = x.recv(4096).decode().split("-")

       print(msg)

       if msg[0] == "::name":

          if msg[1] in comp.conn:

              msg[1]+="#1"

              while msg[1] in comp.conn:

                  msg[1] = msg[1][:-1]+str(int(msg[1][-1])+1)

          print(msg[1])

          u = msg[1]
```

```
        comp.pairList[u] = []

        comp.conn[u] = x

        comp.write(f"::username-{u}",u,True)

        print(420)

        comp.sendall(list(comp.pairList.keys()))

    elif msg[0] == "::cnct":

        p1,p2 = msg[1].split(":")

        if p1 in comp.pairList[p2]:

            print("paired")

            comp.pairList.pop(p1)

            comp.pairList.pop(p2)


            comp.write(f"auth-pair OK-{p2}",p1,True)

            comp.write(f"auth-pair OK-{p1}",p2,True)

        else:

            comp.pairList[p1].append(p2)

    else:

        print(1)

        try:

            print(comp.conn)

            recver = comp.conn[msg[0]]

            comp.write(msg[1],recver)

        except KeyError as e:


            print(e)

            comp.write("client not found", x)

except ConnectionResetError:
    connLeft(x,comp)
def connLeft(x,comp):
    for k in comp.conn:
        if comp.conn[k]==x:
            break
    del comp.conn[k]
    del comp.pairList[k]
    print("client has left: ",k)
    comp.sendall(list(comp.pairList.keys()))

def convertih(ipv4=getIP()):
   return "".join([hex(int(i))[-2:] for i in ipv4.split(".")])



def converthi(ip4_e):
   ip = ""
   for i in range(4):
      if ip4_e[i * 2] == "x":
         ip += str(int("0" + ip4_e[i * 2 + 1], 16))
      else:
         ip += str(int(ip4_e[i * 2 : i * 2 + 2], 16))
      ip += "."
   ip = ip[:-1]
   return ip



t = None
```

```
class Server:
    def __init__(self):
        global t
        print(myip)
        self.socket = sck.socket(sck.AF_INET, sck.SOCK_STREAM)
        self.socket.bind((myip, 6789))
        self.conn = {}
        self.pairList = {}
        t = thr.Thread(target=self.startListen)
        t.start()

    def startListen(self):
        self.socket.listen(5)
        while True:
            x = self.socket.accept()[0]
            t = thr.Thread(target=lambda: read(x,self))
            t.start()

    def write(self, msg,c,user=False):
        print(self.conn)
        if user: c = self.conn[c]
        print(type(msg.encode()))
        c.send(msg.encode())
    def sendall(self,msg,start = "players"):
        print(self.conn)
        for i in self.conn:
            self.write(f"{start}-{str(msg)}",self.conn[i])
class Client:
    def __init__(self, username = None):
        self.socket = sck.socket(sck.AF_INET, sck.SOCK_STREAM)
        self.ip = None
        self.user = username
        self.servconn = []
        self.paired = False
        self.opp = None
        self.eventhand = None
    def read(self):
        while True:
                x = self.socket.recv(4096).decode("utf-8").split("-")
                print(x)
                if x[0] == "players":
                    self.servconn = eval(x[1].lstrip("dict_keys"))
                    print(self.servconn)
                elif x[0]=="::username":
                    self.user = x[1]
                elif x[0]=="auth" and x[1]=="pair OK":
                    self.paired = True
                    self.opp = x[2]
                elif x[0]=="username":
                    self.user = x[1]
                else:self.eventhand(x)#to be imported
```

```
def write(self, msg):
    print(msg.encode())
    self.socket.send(msg.encode())


def setip(self, ip):
    self.ip = converthi(ip)
    print(self.ip)
    try:
        print(self.user)
        self.socket.connect((self.ip, 6789))
        self.write(f"::name-{self.user}")
        t = thr.Thread(target=self.read)
        t.start()
        return True
    except:
        return False
```

**PLAYER MOVE DISTRIBUTON**

```
import pickle

import random as r


class team_creation:

    def __init__(self,root,comp,team):

        #moves-


        print("my team is: ",team)


        print("tema creating")


        final_teams = []


        destruction = [30,50,70,100,130,160]


        #defending

        d
=[['block'],['block','sidepush'],['tackle','sidepush'],['tackle','sidetackle','powerblock'],['supersi
detackle','ultimatetackle','powershoulderpush'],['destructivetackover']]



        #attack


        a =
[['dribble'],['stepover','bodyfeint'],['bodyfeint','rabona'],['rabona','rainbow','elastico'],['rabon
a','elastico','nutmeg'],['ghost body fake','chip','nutmeg','superelastico']]
```

```
        #shooting


        s =
[['powershot','curve'],['placement','curve','head'],['paneka','powershot','topbin'],['powerhea
d','paneka','supershot','swerve']]


        #distribution of moves



        for i in range(len(team)):
            player_moves=[]
            for x in range(1,len(team[i])):
                if team[i][x][2] == 'LM' or team[i][x][2] == 'RM' or team[i][x][2] == 'CAM' or
team[i][x][2] == 'RW' or team[i][x][2] == 'ST' or team[i][x][2] == 'LW' or team[i][x][2] == 'CF':
                    if team[i][x][1] <= 100 and team[i][x][1] >=80:
                        l1 = r.randint(0,3)
                        l2 = r.randint(0,2)
                        l3 = r.randint(0,3)

                        m1=a[5][l1]
                        m2=a[4][l2]
                        s1 = s[3][l3]

                        l = [team[i][x][0],team[i][x][2],[m1,160],[m2,130],[s1,160],200]
                        player_moves.append(l)



                    elif team[i][x][1] < 80 and team[i][x][1] >=70:
```

```
   l1 = r.randint(0,2)

   l2 = r.randint(0,2)

   l3 = r.randint(0,2)


   m1=a[4][l1]

   m2=a[3][l2]

   s1 = s[2][l3]


   l = [team[i][x][0],team[i][x][2],[m1,130],[m2,100],[s1,130],160]

   player_moves.append(l)



elif team[i][x][1] < 70 and team[i][x][1] >= 50:

   l1 = r.randint(0,2)

   l2 = r.randint(0,1)

   l3 = r.randint(0,2)


   m1=a[3][l1]

   m2=a[2][l2]

   s1 = s[1][l3]


   l = [team[i][x][0],team[i][x][2],[m1,100],[m2,70],[s1,100],100]

   player_moves.append(l)



elif team[i][x][1] < 50:

   l1 = r.randint(0,1)
```

```
                    l3 = r.randint(0,1)


                    m1=a[1][l1]

                    m2=a[0]

                    s1 = s[0][l3]


                    l = [team[i][x][0],team[i][x][2],[m1,50],[m2,30],[s1,70],70]

                    player_moves.append(l)



                elif team[i][x][2] == 'LB' or team[i][x][2] == 'RB' or team[i][x][2] == 'CDM' or
team[i][x][2] == 'CM' or team[i][x][2] == 'GK' or team[i][x][2] == 'CB' :

                  if team[i][x][1] <= 100 and team[i][x][1] >=80:

                    l2 = r.randint(0,2)

                    l3 = r.randint(0,3)


                    m1=d[5][0]

                    m2=d[4][l2]

                    s1 = s[3][l3]


                    l = [team[i][x][0],team[i][x][2],[m1,160],[m2,130],[s1,160],200]

                    player_moves.append(l)



                  elif team[i][x][1] < 80 and team[i][x][1] >=70:

                    l1 = r.randint(0,2)

                    l2 = r.randint(0,2)

                    l3 = r.randint(0,2)
```

```
    m1=d[4][l1]

    m2=d[3][l2]

    s1 = s[2][l3]


    l = [team[i][x][0],team[i][x][2],[m1,130],[m2,100],[s1,130],160]

    player_moves.append(l)



  elif team[i][x][1] < 70 and team[i][x][1] >=50:

    l1 = r.randint(0,2)

    l2 = r.randint(0,1)

    l3 = r.randint(0,2)


    m1=d[3][l1]

    m2=d[2][l2]

    s1 = s[1][l3]


    l = [team[i][x][0],team[i][x][2],[m1,100],[m2,70],[s1,100],100]

    player_moves.append(l)



  elif team[i][x][1] < 50:

    l1 = r.randint(0,1)

    l3 = r.randint(0,1)


    m1=d[1][l1]
```

```
            m2=d[0]

            s1 = s[0][l3]


            l = [team[i][x][0],team[i][x][2],[m1,50],[m2,30],[s1,70],70]

            player_moves.append(l)


        player_moves.insert(0,team[i][0])

        final_teams.append(player_moves)



with open("team.dat","wb") as file:

    pickle.dump(final_teams,file)
```

**DATALOADING**

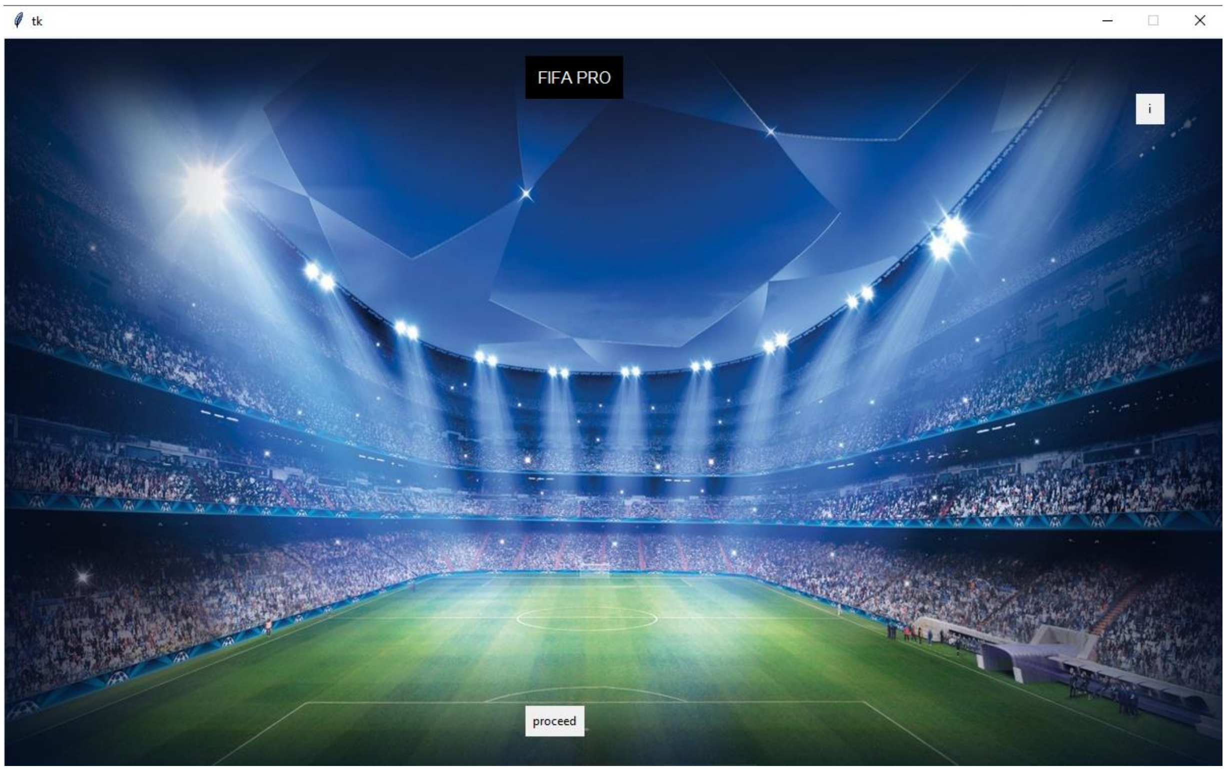
```
import csv
import mysql.connector as sql
import threading as thr
def maketable(filename,passw,primary_key = "Name",additional = ")"):
    do = sql.connect(host = "localhost", user = "root",password = passw)
    ci = do.cursor()
    ci.execute("create database if not exists fifadata")
    ci.execute("use fifadata")
    csvr = csv.reader(open(filename+".csv",encoding='utf8'))
    indexes = next(csvr)
    sample = next(csvr)
    s = f"create table if not exists {filename}("
    for i in range(len(indexes)):
        if sample[i].isdigit():
            s+=f"{indexes[i]} int"
        elif sample[i].lower() in ["true","false"]:
            s+=f"{indexes[i]} boolean"
        else:
            s+=f"{indexes[i]} varchar(100)"
        if indexes[i]==primary_key:
            s+="primary key"
        s+=","
    s = s[:-1]
    s+=additional
    print(s)
    ci.execute(s)
    print(f"table {filename} successfully created")
    do.commit()
def loadData(filename,passw):
    do = sql.connect(host = "localhost", user = "root",password = passw)
    ci = do.cursor()
    ci.execute("use fifadata")
    ci.execute(f"delete from {filename}")
    ci.execute(f"desc {filename}")
    datatypes = []
    for i in ci:
        datatypes.append(i[1])
    print(datatypes)
    csvr = csv.reader(open(filename+".csv",encoding = "utf8"))
    next(csvr)
    for i in csvr:
        s = f"insert into {filename} values("
        for j in range(len(datatypes)):
            if datatypes[j] == b'int':
                if i[j] == "":
                    s+="0"
                else:
                    s+=f"{eval(i[j])}"
            elif datatypes[j] ==b'tinyint(1)':
                s+=f"{bool(i[j])}"
            else:
                s+=f'"{i[j]}"'
            s+=","
        s = s[:-1]
        s+=")"
        try:
            ci.execute(s)
        except sql.Error as e:
            pass
        do.commit()
    print(f"data loaded into {filename}")
```

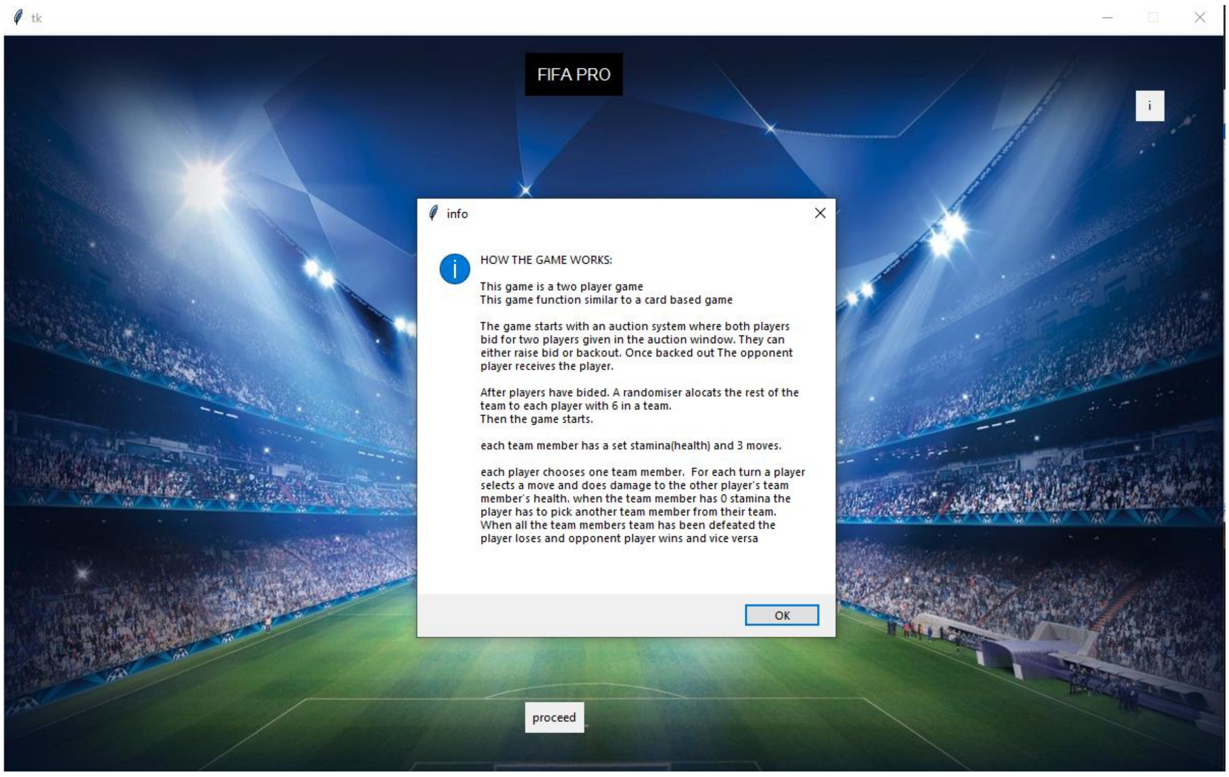
**AUCTION SYSTEM**

```
import mysql.connector as sql

import random as rand

def pickAuctionPlayer(passw,filename = "players_fifa22", primary_key = "name",ovr =
"potential",size = 50,influence = 2):

    do = sql.connect(host = "localhost", user = "root", password = passw, database = "fifadata")

    ci = do.cursor()

    ci.execute(f"desc {filename}")

    datatypes = []

    for i in ci:

        datatypes.append(i[0].lower())

    ci.execute(f"select*from {filename} order by potential desc limit {size}")

    p = list(ci)

    ovrList= []

    for i in p:

        ovrList.append(i[datatypes.index(ovr)]**2)

    pick = rand.random()*sum(ovrList)

    for i in range(len(ovrList)):

        if pick-ovrList[i]>=0:

            pick-=ovrList[i]

        else:

            print(i)

            return p[i]

    print(-1)

    return p[-1]

def team_init(luck,passw,filename = "players_fifa22"):

    pl = []
```

```
op = []
do = sql.connect(host = "localhost", user = "root", password = passw, database = "fifadata")
ci = do.cursor()
formn = [3,3]
pos = [["LB","RB","CB","CDM","CM","GK"],["LM","RM","CAM","RW","ST","LW","CF"]]
for k in range(len(formn)):
  s = "select * from players_fifa22 where"
  for p in pos[k]:
    s+=f" bestposition = '{p}' or"
  s = s[:-2]+f"order by rand() limit 50"
  ci.execute(s)
  temp=[]
  line = []
  for j in range(formn[k]):
    pool = 0
    for p in ci:
      temp.append( p)
      pool+=p[1]**luck
    num = rand.random()*pool
    for i in range(len(temp)):
      if num-temp[i][1]**luck>=0:
        num-=temp[i][1]**luck
      else:
        print(i)
        t=temp.pop(i)
        break
    else:
            t = temp.pop(len(temp)-1)
        line.append(t)
    pl.append(line)
for j in pl:
    for i in j:
        print(i[0],i[1],end = "\t")
    print()
return pl
```

# OUTPUT





HOW THE GAME WORKS:

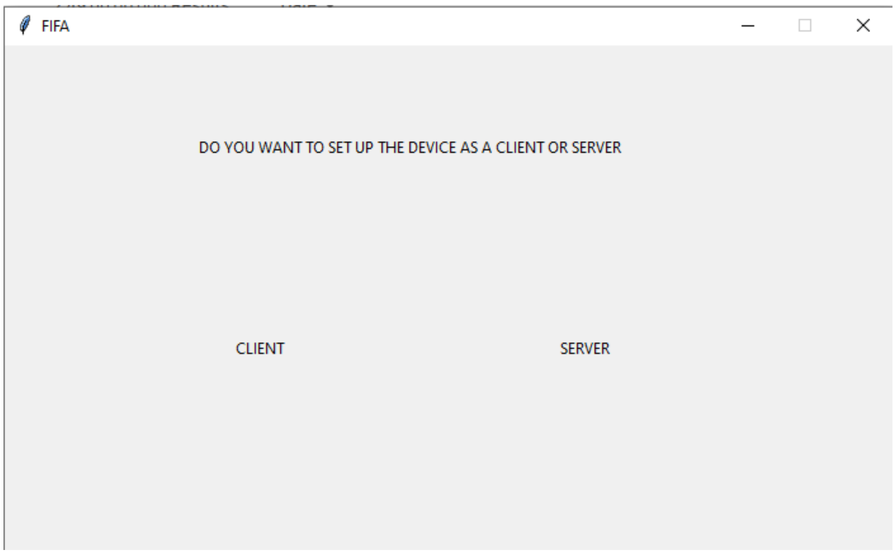
This game is a two player game
This game function similar to a card based game

The game starts with an auction system where both players bid for two players given in the auction window. They can either raise bid or backout. Once backed out The opponent player receives the player.

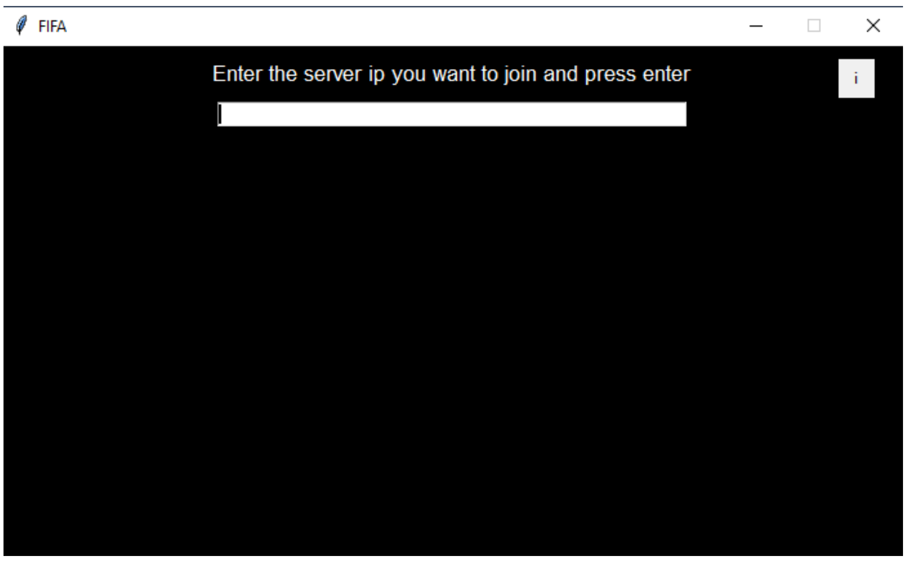
After players have bided. A randomiser alocats the rest of the team to each player with 6 in a team.
Then the game starts.

each team member has a set stamina(health) and 3 moves.

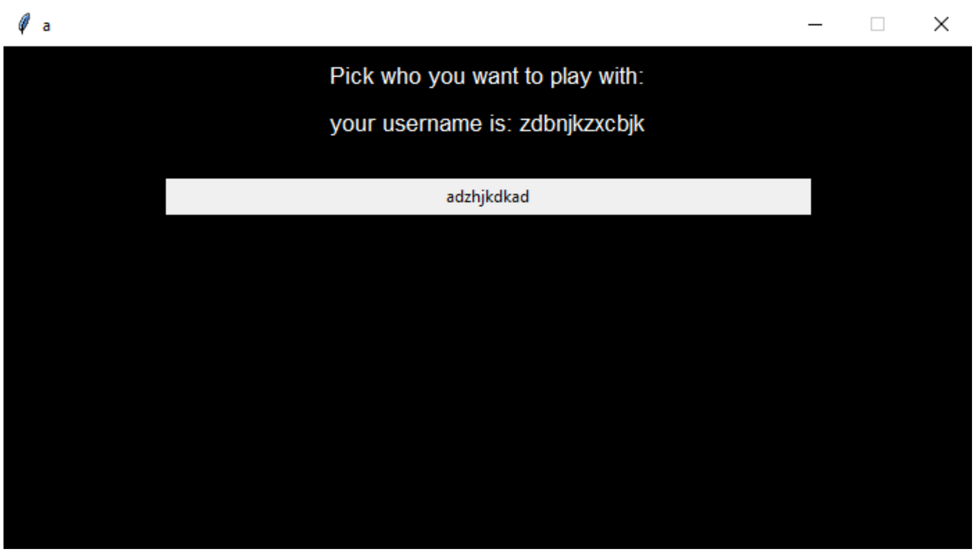
each player chooses one team member.  For each turn a player selects a move and does damage to the other player's team member's health. when the team member has 0 stamina the player has to pick another team member from their team.
When all the team members team has been defeated the player loses and opponent player wins and vice versa



DO YOU WANT TO SET UP THE DEVICE AS A CLIENT OR SERVER

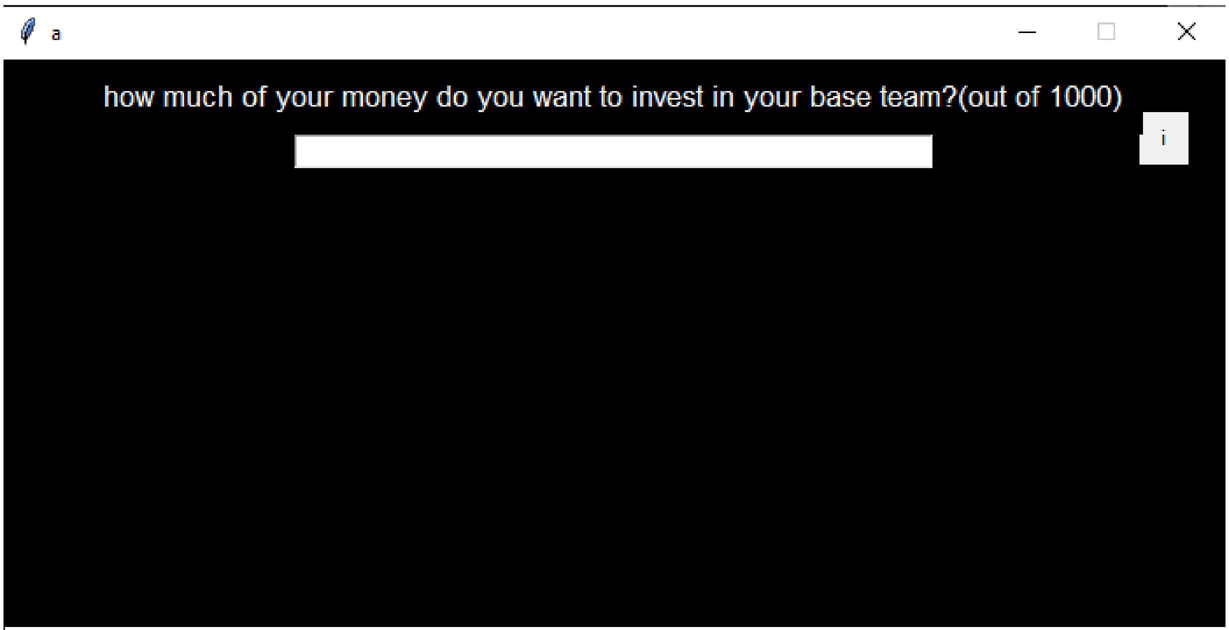


Enter the server ip you want to join and press enter

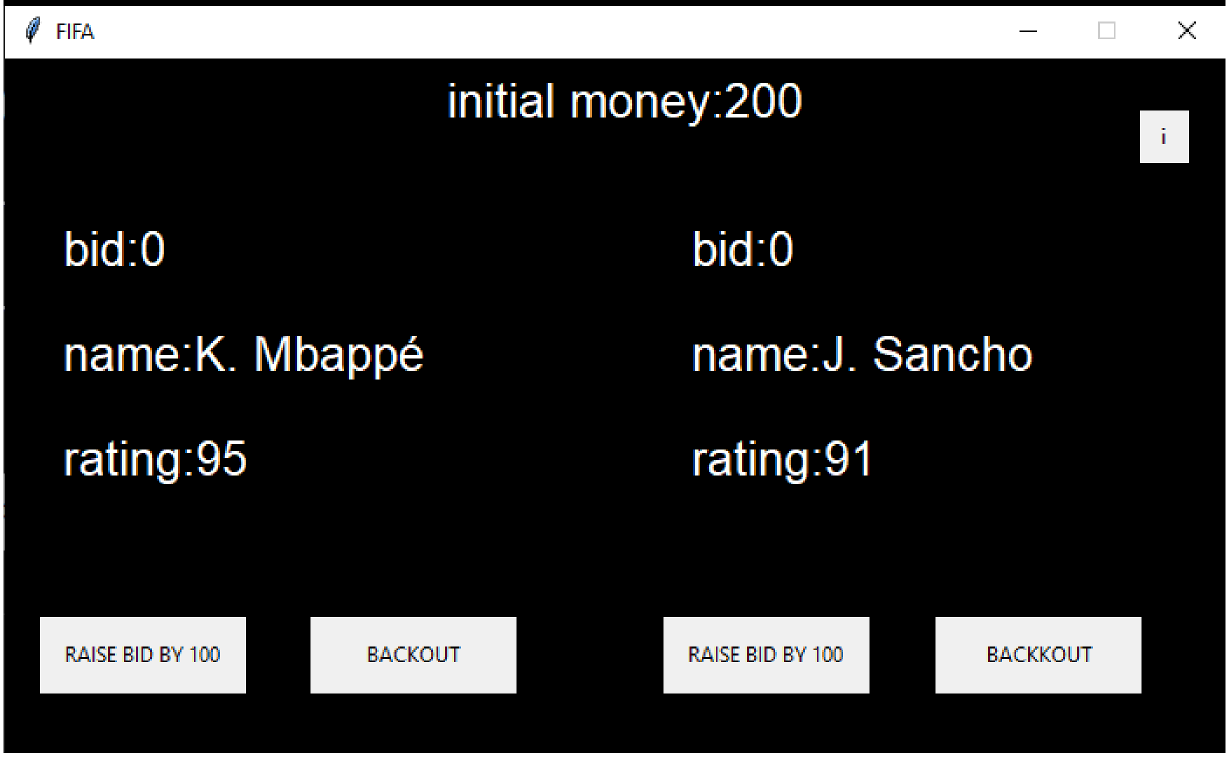


Pick who you want to play with:

your username is: zdbnjkzxcbjk



how much of your money do you want to invest in your base team?(out of 1000)



initial money:200

bid:0

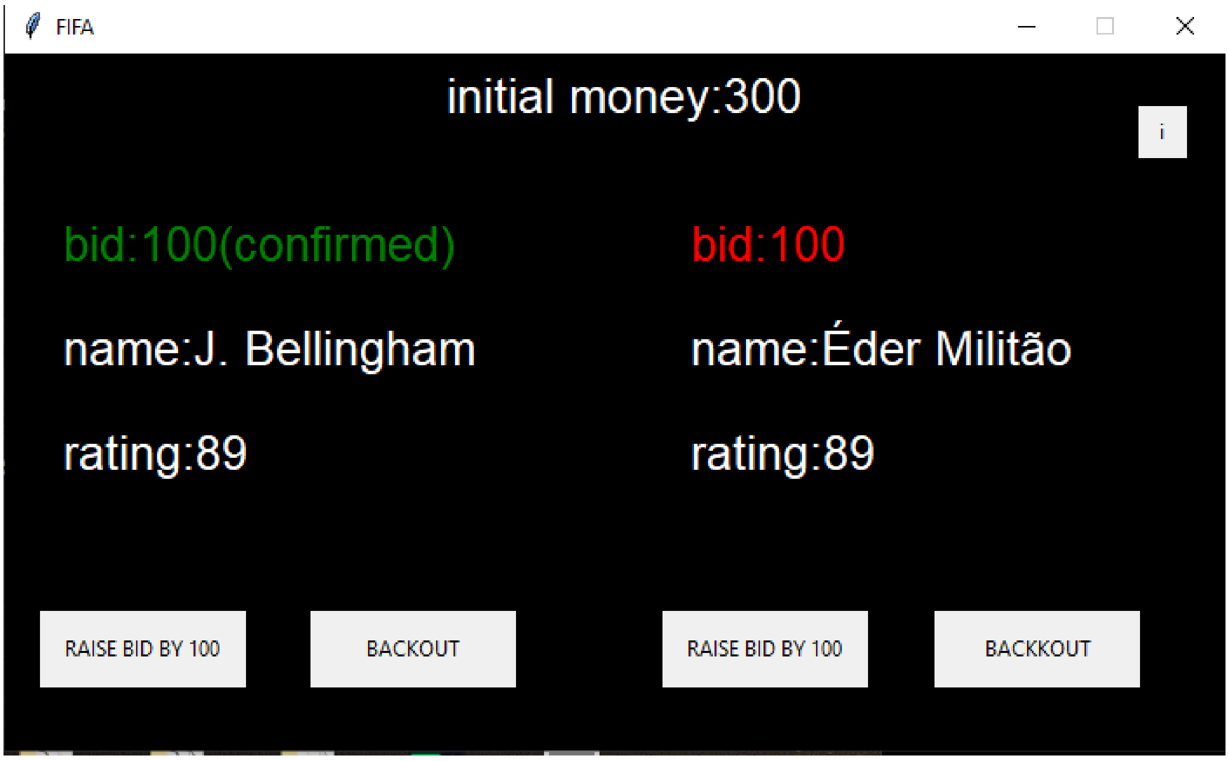
name:K. Mbappé

rating:95

bid:0

name:J. Sancho

rating:91


FIFA

initial money:300

i

bid:100(confirmed)

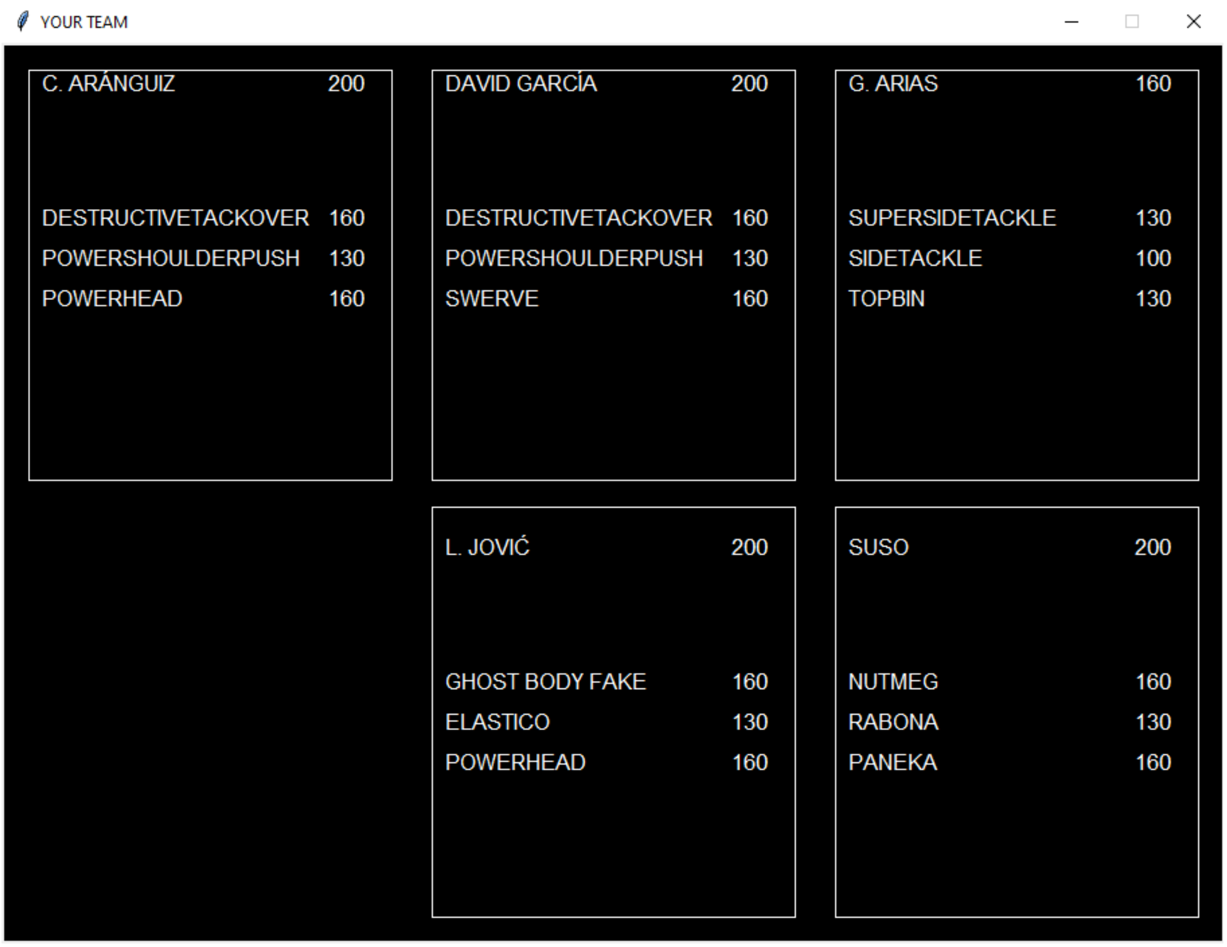
name:J. Bellingham

rating:89

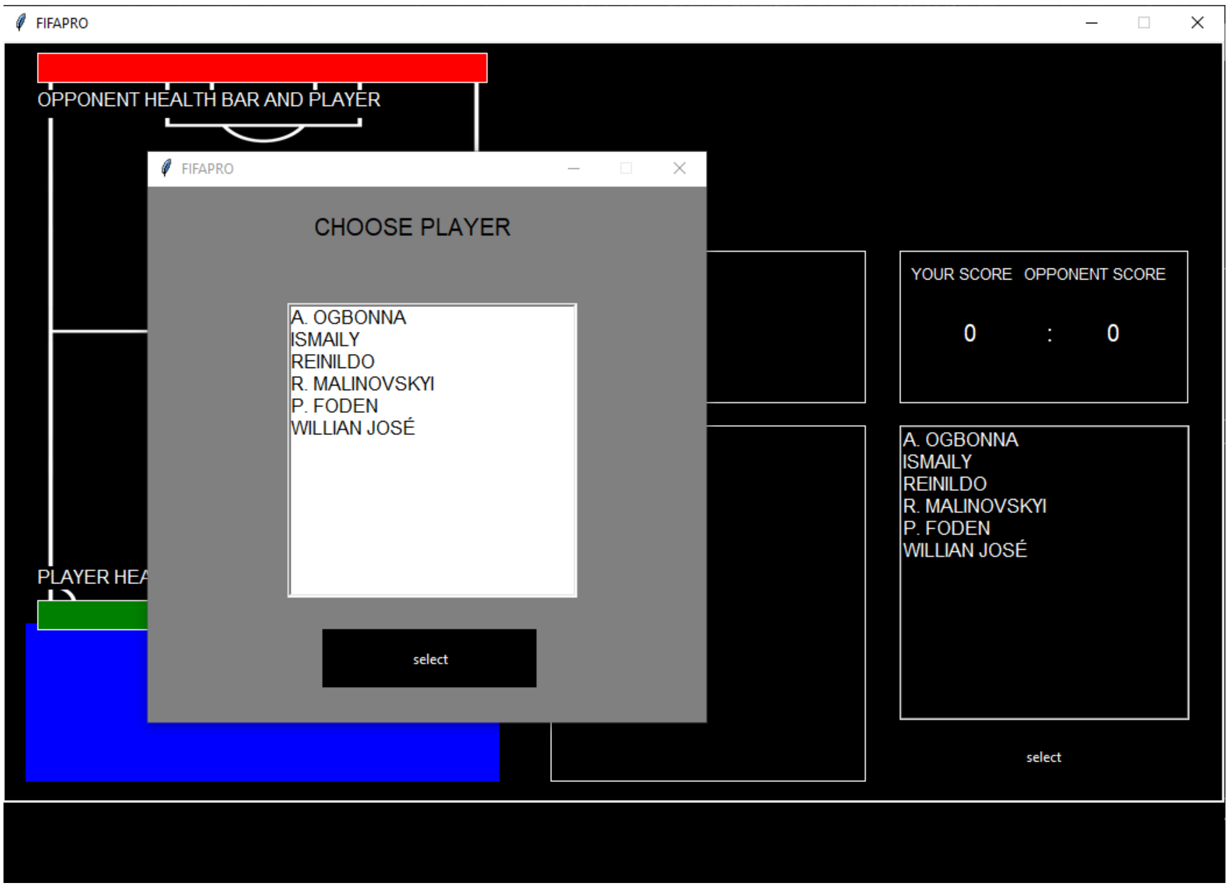
bid:100

name:Éder Militão

rating:89


YOUR TEAM

```
C. ARÁNGUIZ                200
DESTRUCTIVETACKOVER        160
POWERSHOULDERPUSH          130
POWERHEAD                  160

DAVID GARCÍA               200
DESTRUCTIVETACKOVER        160
POWERSHOULDERPUSH          130
SWERVE                     160

G. ARIAS                   160
SUPERSIDETACKLE            130
SIDETACKLE                 100
TOPBIN                     130

L. JOVIĆ                   200
GHOST BODY FAKE            160
ELASTICO                   130
POWERHEAD                  160

SUSO                       200
NUTMEG                     160
RABONA                     130
PANEKA                     160
```


FIFAPRO

OPPONENT HEALTH BAR AND PLAYER

FIFAPRO

CHOOSE PLAYER

A. OGBONNA
ISMAILY
REINILDO
R. MALINOVSKYI
P. FODEN
WILLIAN JOSÉ

select

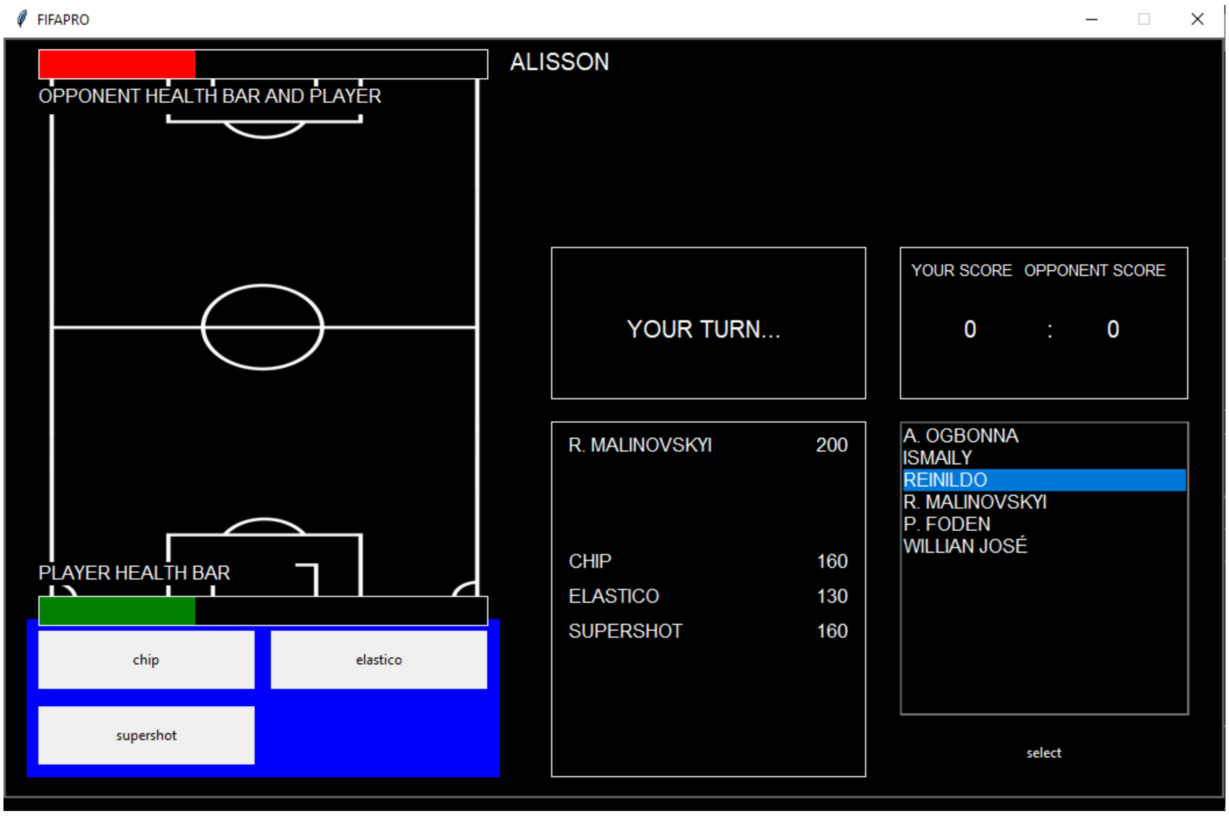
PLAYER HEA

YOUR SCORE OPPONENT SCORE

0 : 0

A. OGBONNA
ISMAILY
REINILDO
R. MALINOVSKYI
P. FODEN
WILLIAN JOSÉ

select


FIFAPRO

OPPONENT HEALTH BAR AND PLAYER

ALISSON

YOUR TURN...

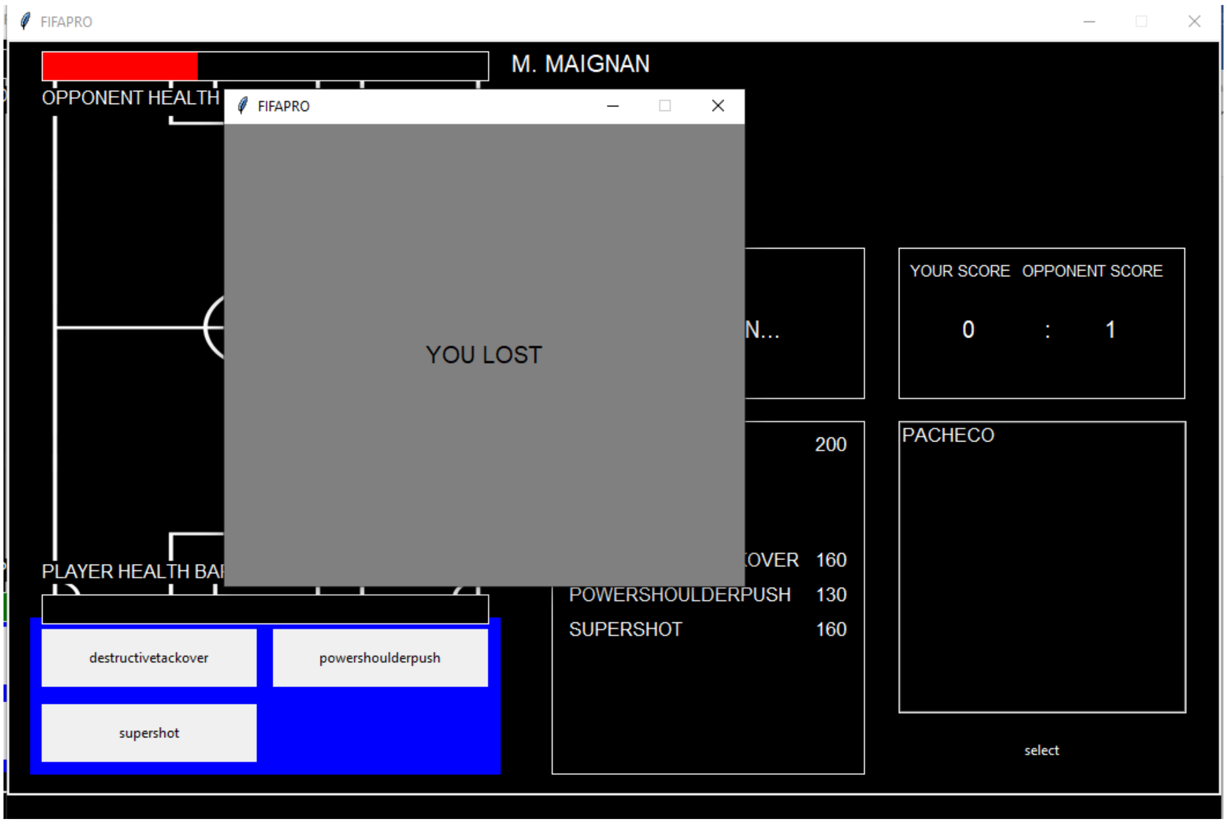
YOUR SCORE OPPONENT SCORE

0 : 0

```
R. MALINOVSKYI      200

CHIP                160
ELASTICO            130
SUPERSHOT           160
```

A. OGBONNA
ISMAILY
REINILDO
R. MALINOVSKYI
P. FODEN
WILLIAN JOSÉ

PLAYER HEALTH BAR


FIFAPRO

M. MAIGNAN

OPPONENT HEALTH

FIFAPRO

YOU LOST

N...

YOUR SCORE OPPONENT SCORE

0 : 1

200

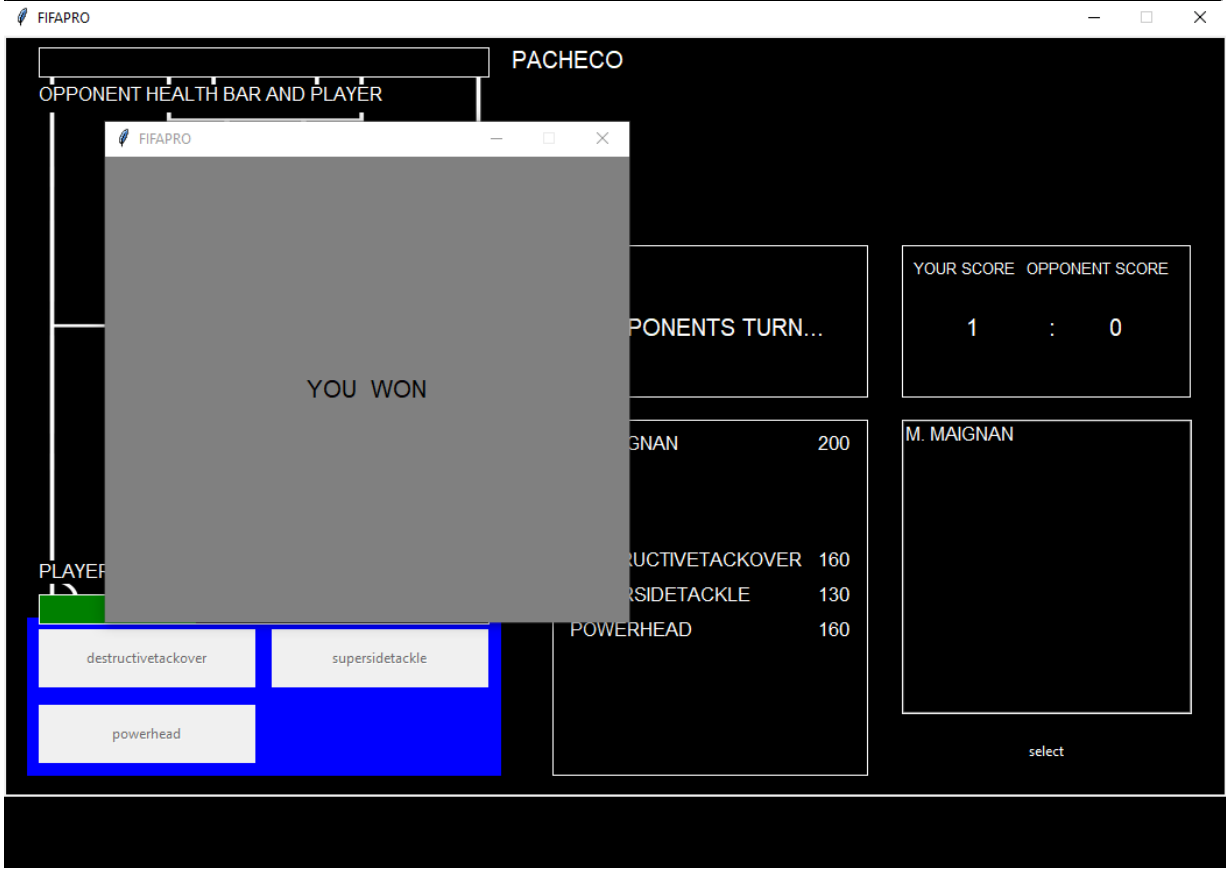
PACHECO

OVER 160

POWERSHOULDERPUSH 130

SUPERSHOT 160

PLAYER HEALTH BAR


FIFAPRO

PACHECO

OPPONENT HEALTH BAR AND PLAYER

FIFAPRO

YOU WON

PONENTS TURN...

YOUR SCORE OPPONENT SCORE

1 : 0

GNAN 200

M. MAIGNAN

UCTIVETACKOVER 160

RSIDETACKLE 130

POWERHEAD 160

PLAYER

## LIMITATIONS

1. The game supports inter system connectivity only if the 2 systems are close to each other.
2. The connection process doesn't show the available servers in the user's vicinity
3. The friendliness of the GUI can be enhanced
4. Only limited information about the player is shown.
5. The GUI doesn't have a means to show the image of the player.
6. The game makes a very limited use of the player's statistics.

# BIBLIOGRAPHY

1. Socket Programming in Python - GeeksforGeeks
2. socket — Low-level networking interface — Python 3.10.6 documentation
3. FIFA 20 complete player dataset | Kaggle