# STONE PAPER SCISSOR GAME USING SOCKET PROGARMMING

A COURSE PROJECT REPORT

By

**P. VINAY (RA2011030010138)**
**MADAVARAM REDDI VINATHI(RA2011030010133)**
**GUVVALA SAI AMARNADH SATVIC REDDY(RA2011030010124)**

Under the guidance of

**<PRASATH N>**

*In partial fulfilment for the Course*

of

18CSC302J - COMPUTER NETWORKS

in <NWC>


**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Kattankulathur, Chenpalpattu District**

NOVEMBER 2022

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

**(Under Section 3 of UGC Act, 1956)**

## BONAFIDE CERTIFICATE

Certified that this mini project report "**Stone paper scissor game using socket programming** " is the bonafide work of **Satvic (RA2011030010124)** , **MR Vinathi (RA2011030010133) and P Vinay (RA2011030010138)** who carried out the project work under my supervision.

**SIGNATURE**

<Faculty Name>

**<Designation>**

**<Department>**
SRM Institute of Science and Technology

**ABSTRACT**

In this project, we will be implementing a rock-paper-scissors game! using socket. Rock-paper-scissors is a hand game played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "rock" (a simple fist), "paper" (a flat hand), and "scissors" (a fist with the index and middle fingers extended, forming a V). A player who plays rock will beat another player who has chosen scissors, but will lose to one who has played paper; a play of paper will lose to a play of scissors. If both players choose the same shape, the game is tied.

## ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **<Faculty Name> , <Designation> , <Department>,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD <Name> <Designation>, <Department>** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

# TABLE OF CONTENTS

# 1. INTRODUCTION

## 1.1 Scenario Description

Now-a-days, everyone know of how to play rock paper scissor game. But ever thought of playing that game in online. So we came up with this project which allows the users play the game with their own device.

Most of us may have played rock paper scissors before. Rock Paper Scissors is often used as a fair selection method between two people or more in order to settle a dispute or make an unbiased group decision. This method is similar to flipping a coin, drawing straws, or throwing dice.

We have to setup a server which listens on Port 8080 (HTTPS) only. The private IP address should be marked by user in the server and client side so that it can run in their own device. In this way the user will come to know about his selection so that it will be fair for both users to play the game.

# 2. LITERATURE SURVEY

Almost everyone have settled disputes by playing a simple game called Paper-Rock-Scissor (PRS). The rule for winning a one-shot play of this game is simple: rock crushes scissors, scissors cut paper, and paper covers rock. But in addition to being a fun game to resolve disagreements, RPS is also a serious game used by game theorists and psychologists to study competitive behavior in naturalistic settings, such as security, terrorism and war (Fisher, 2008). Because none of the strategies (P, R, or S) is absolutely better than either of the other two, PRS is an interesting research paradigm to study adversarial behavior and dynamics of player's strategies. For example, rock can beat paper but at the same time rock can be beaten by scissors. However, most previous work focused on one-shot games involving models and humans or two humans but with re-shuffling of the players rather than observing repeated plays of the same pair (but see Hoffman, Suetens, Gneezy, & Nowak, 2015, for exceptions).

In the traditional form of the PRS game a win gives 1 point to the winner and takes 1 point from the loser and a tie gives 0 points to both players. With this setting, a player who plays RPS randomly has a 1/3 chance of winning in any round. Importantly, this is also the optimal strategy — i.e., Nash Equilibrium strategy (the strategy where no player's deviation is beneficial): in each trial, if player 1 chooses each strategy 1/3 of the time, player 1's payoff is 0 regardless of player 2's strategy. Otherwise, player 2 could find a combination strategies that would win the game with a positive expected value. Thus, in all previous behavioral research with this traditional zero-sum symmetrical payoff design, it is not possible to distinguish whether players are in agreement to Nash or random strategy. Some researchers have found evidence of strategies that are consistent with Nash, suggesting that experienced players who use information of previous plays strategically are more likely to win (Batzilis, Jaffe, Levitt, List, & Picel, 2019).

# 3. REQUIREMENTS

## 3.1 Requirement Analysis

From the given scenario, we draw the following requirements:

* Users on the internet should be able to access only https on the e-commerce server.

* Users on the internet should have access only to the public IP address of the server and not the private IP address.

* The users in the organization should have full access to the server.

* Features and configuration required on to run the application:

* Make sure you have python installed and setup on your system. App is tested on Python 2.7

* To start the server: python game_server.py

* Click "Start" button on the Server window

* Launch two clients. To start a client: python game_client.py

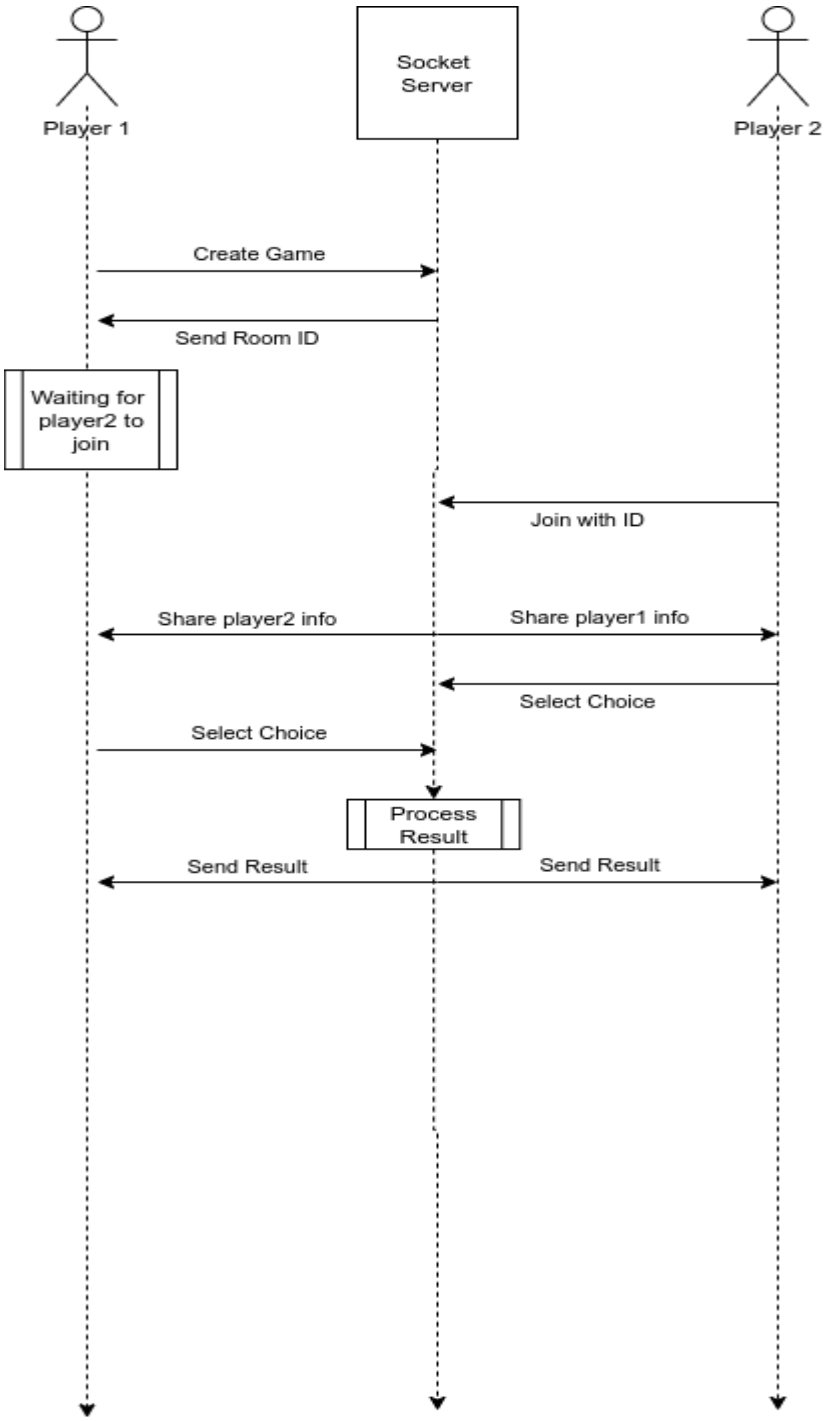
* Enter player name and click on "Connect" button

* The game starts when two clients (players) are connected.

* Enjoy!

# 4. ARCHITECTURE AND DESIGN

## 4.1 Game Flow




This is a sequence diagram that shows the timeline of events.

# 5. IMPLEMENTATION

## 5.1 Module 1:

- Import important libraries
- Define variables and dictionaries
- Create the overall layout of the Python Rock Paper Scissors Project
- Create important functions
- Import important libraries

**Code**:

```
import socket
import threading
import tkinter as tk
from time import sleep
```

**Explanation:**

First, we import the tkinter library, a widely used GUI library, after that we import random. random is basically used to generate a random number from a given range.

```
server = None
HOST_ADDR = "192.168.222.1"
HOST_PORT = 8080
client_name = " "
clients = []
clients_names = []
player_data = []
```

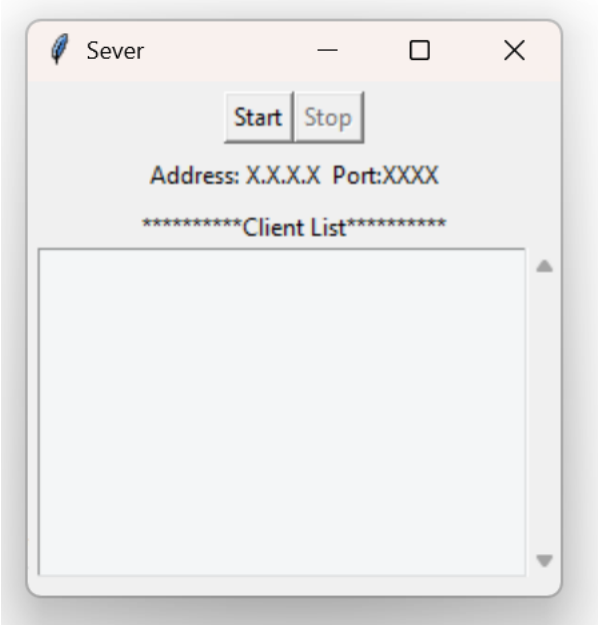
**Explanation:**

Connecting to the client from server using device's own IPV4 address and standard port number("8080") .

# 6. RESULTS AND DISCUSSION

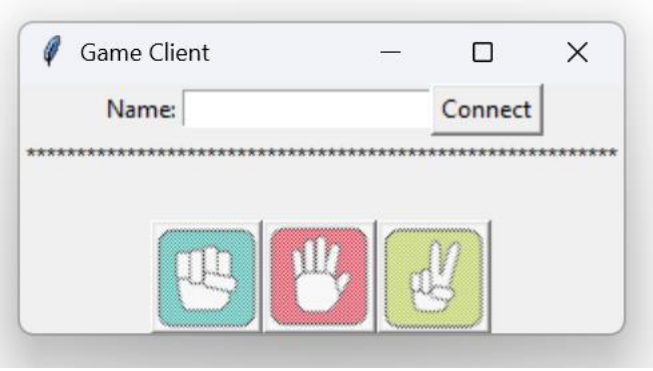
## 6.1 Connection Check

**Connection establishment from server side:**



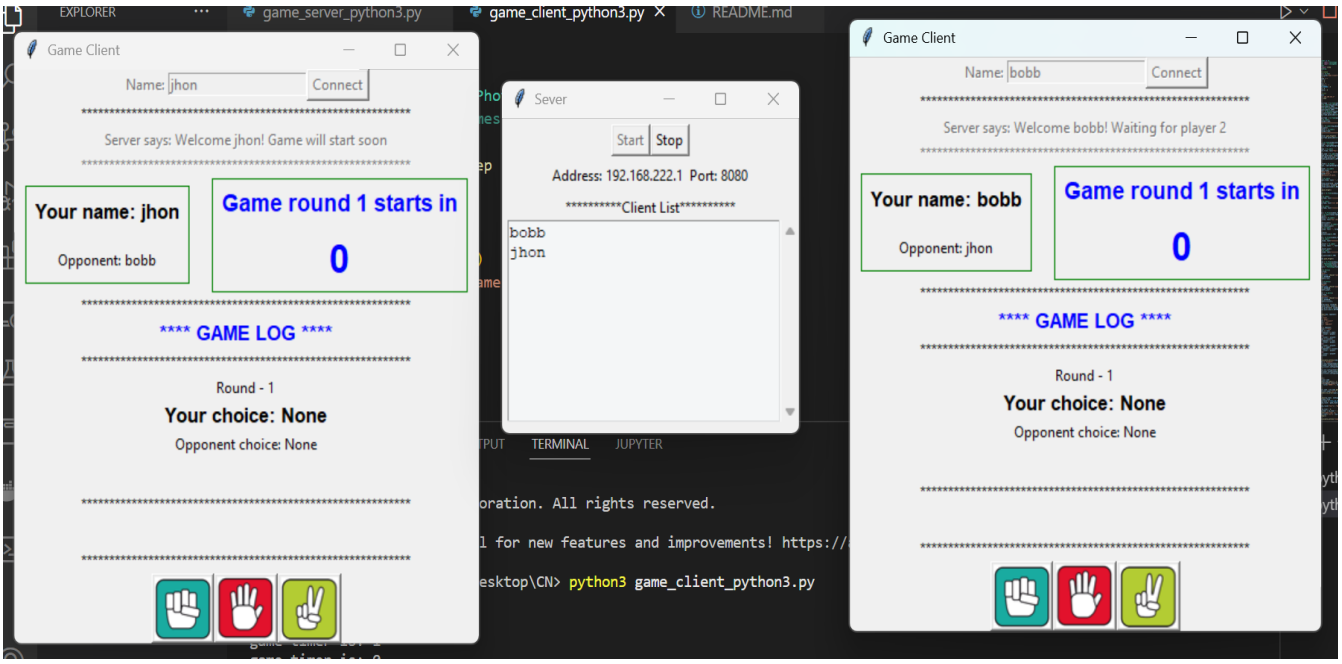

This tkinter window shows that user is successfully connected to the server under the port 8080 and his device's own IP adress .

**Connection establishment from client side:**

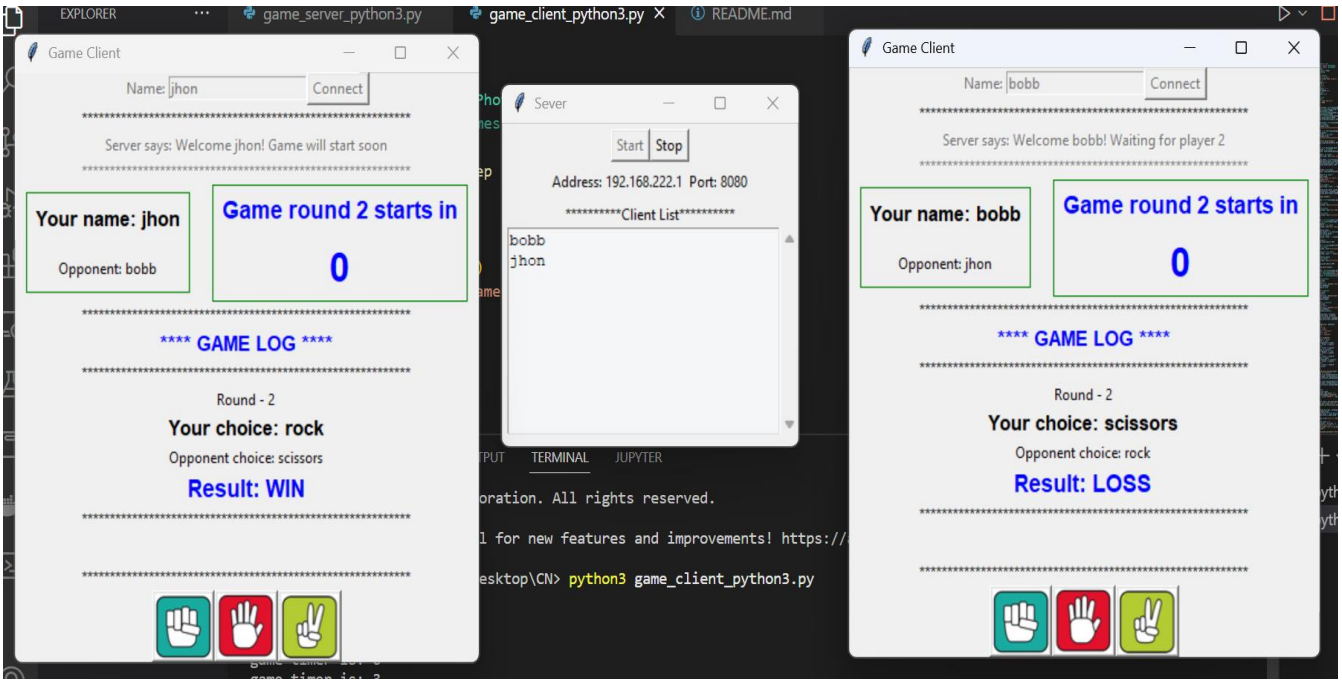



This tkinter window shows that the client also successfully connected to the server under device's own IP adress and a successful client-server connection established.

**Game Execution:**




Enter names(lets say “Jhon” and “Bobb”) so that server can register your names.And as the game round starts, choose whatever you want ( rock or scissor or paper).




After choosing , the srever will register your choices and checks with code and decides the winner.

## 7. CONCLUSION AND FUTURE ENHANCEMENT

So, this was an easy and fun way to create a rock paper scissors game. It is customizable, as per a developer's personal preference. Not just rock paper scissors, but many more games can be developed easily in Python using various tools and libraries available.

# REFERENCES

https://www.geeksforgeeks.org/rock-paper-and-scissor-game-using-tkinter/

https://www.tutorialspoint.com/rock-paper-and-scissor-game-using-tkinter

## CODES:

**game_server_python3.py**

```
import socket
import threading
import tkinter as tk
from time import sleep
window = tk.Tk()
window.title("Sever")
# Top frame consisting of two buttons widgets (i.e. btnStart, btnStop)
topFrame = tk.Frame(window)
btnStart = tk.Button(topFrame, text="Start", command=lambda: start_server())
btnStart.pack(side=tk.LEFT)
btnStop = tk.Button(
    topFrame, text="Stop", command=lambda: stop_server(), state=tk.DISABLED)
btnStop.pack(side=tk.LEFT)
topFrame.pack(side=tk.TOP, pady=(5, 0))
# Middle frame consisting of two labels for displaying the host and port info
middleFrame = tk.Frame(window)
lblHost = tk.Label(middleFrame, text="Address: X.X.X.X")
lblHost.pack(side=tk.LEFT)
lblPort = tk.Label(middleFrame, text="Port:XXXX")
lblPort.pack(side=tk.LEFT)
middleFrame.pack(side=tk.TOP, pady=(5, 0))
# The client frame shows the client area
clientFrame = tk.Frame(window)
lblLine = tk.Label(clientFrame, text="**********Client List**********").pack()
scrollBar = tk.Scrollbar(clientFrame)
scrollBar.pack(side=tk.RIGHT, fill=tk.Y)
tkDisplay = tk.Text(clientFrame, height=10, width=30)
tkDisplay.pack(side=tk.LEFT, fill=tk.Y, padx=(5, 0))
scrollBar.config(command=tkDisplay.yview)
tkDisplay.config(
    yscrollcommand=scrollBar.set,
```

```
    background="#F4F6F7",
    highlightbackground="grey",
    state="disabled",)
clientFrame.pack(side=tk.BOTTOM, pady=(5, 10))
server = None
HOST_ADDR = "192.168.222.1"
HOST_PORT = 8080
client_name = " "
clients = []
clients_names = []
player_data = []
# Start server function
def start_server():
    global server, HOST_ADDR, HOST_PORT  # code is fine without this
    btnStart.config(state=tk.DISABLED)
    btnStop.config(state=tk.NORMAL)
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print(socket.AF_INET)
    print(socket.SOCK_STREAM)
    server.bind((HOST_ADDR, HOST_PORT))
    server.listen(5)  # server is listening for client connection
    threading._start_new_thread(accept_clients, (server, " "))
    lblHost["text"] = "Address: " + HOST_ADDR
 lblPort["text"] = "Port: " + str(HOST_PORT)
# Stop server function
def stop_server():
    global server
    btnStart.config(state=tk.NORMAL)
    btnStop.config(state=tk.DISABLED)
def accept_clients(the_server, y):
    while True:
        if len(clients) < 2:
            client, addr = the_server.accept()
            clients.append(client)
            # use a thread so as not to clog the gui thread
```

```
            threading._start_new_thread(send_receive_client_message, (client, addr))
    # Function to receive message from current client AND
    # Send that message to other clients
    def send_receive_client_message(client_connection, client_ip_addr):
        global server, client_name, clients, player_data, player0, player1
        client_msg = " "
        # send welcome message to client
        client_name = client_connection.recv(4096).decode()
        if len(clients) < 2:
            client_connection.send("welcome1".encode())
        else:
            client_connection.send("welcome2".encode())
        clients_names.append(client_name)
        update_client_names_display(clients_names)  # update client names display
        if len(clients) > 1:
            sleep(1)
            # send opponent name
            clients[0].send(("opponent_name$" + clients_names[1]).encode())
            clients[1].send(("opponent_name$" + clients_names[0]).encode())
            # go to sleep
        while True:
            data = client_connection.recv(4096).decode()
            if not data:
                break
            # get the player choice from received data
            player_choice = data[11 : len(data)]
            msg = {"choice": player_choice, "socket": client_connection}
            if len(player_data) < 2:
                player_data.append(msg)
            if len(player_data) == 2:
                # send player 1 choice to player 2 and vice versa
                dataToSend0 = "$opponent_choice" + player_data[1].get("choice")
                dataToSend1 = "$opponent_choice" + player_data[0].get("choice")
                player_data[0].get("socket").send(dataToSend0.encode())
                player_data[1].get("socket").send(dataToSend1.encode())
```

```
            player_data = []
    # find the client index then remove from both lists(client name list and connection list)
    idx = get_client_index(clients, client_connection)
    del clients_names[idx]
    del clients[idx]
    client_connection.close()

    update_client_names_display(clients_names)  # update client names display
# Return the index of the current client in the list of clients
def get_client_index(client_list, curr_client):
    idx = 0
    for conn in client_list:
        if conn == curr_client:
            break
        idx = idx + 1
    return idx
```

**game_client_python3.py**

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
import socket
from time import sleep
import threading
# MAIN GAME WINDOW
window_main = tk.Tk()
window_main.title("Game Client")
your_name = ""
opponent_name = ""
game_round = 0
game_timer = 4
your_choice = ""
opponent_choice = ""
TOTAL_NO_OF_ROUNDS = 5
your_score = 0
opponent_score = 0
```

```
# network client
client = None
HOST_ADDR = "192.168.222.1"
HOST_PORT = 8080
top_welcome_frame = tk.Frame(window_main)
lbl_name = tk.Label(top_welcome_frame, text="Name:")
lbl_name.pack(side=tk.LEFT)
ent_name = tk.Entry(top_welcome_frame)
ent_name.pack(side=tk.LEFT)
btn_connect = tk.Button(top_welcome_frame, text="Connect", command=lambda: connect())
btn_connect.pack(side=tk.LEFT)
top_welcome_frame.pack(side=tk.TOP)
top_message_frame = tk.Frame(window_main)
lbl_line = tk.Label(
    top_message_frame,
    text="************************************************************",).pack()
lbl_welcome = tk.Label(top_message_frame, text="")
lbl_welcome.pack()
lbl_line_server = tk.Label(
    top_message_frame,
    text="************************************************************")
lbl_line_server.pack_forget()
top_message_frame.pack(side=tk.TOP)
top_frame = tk.Frame(window_main)
top_left_frame = tk.Frame(
    top_frame, highlightbackground="green", highlightcolor="green", highlightthickness=1)
lbl_your_name = tk.Label(
    top_left_frame, text="Your name: " + your_name, font="Helvetica 13 bold")
lbl_opponent_name = tk.Label(top_left_frame, text="Opponent: " + opponent_name)
lbl_your_name.grid(row=0, column=0, padx=5, pady=8)
lbl_opponent_name.grid(row=1, column=0, padx=5, pady=8)
top_left_frame.pack(side=tk.LEFT, padx=(10, 10))
top_right_frame = tk.Frame(
    top_frame, highlightbackground="green", highlightcolor="green", highlightthickness=1)
lbl_game_round = tk.Label(
```

```
    top_right_frame,
    text="Game round (x) starts in",
    foreground="blue",
    font="Helvetica 14 bold",)
lbl_timer = tk.Label(
    top_right_frame, text=" ", font="Helvetica 24 bold", foreground="blue")
lbl_game_round.grid(row=0, column=0, padx=5, pady=5)
lbl_timer.grid(row=1, column=0, padx=5, pady=5)
top_right_frame.pack(side=tk.RIGHT, padx=(10, 10))
top_frame.pack_forget()
middle_frame = tk.Frame(window_main)
lbl_line = tk.Label(
    middle_frame,
text="************************************************************").pack()
lbl_line = tk.Label(
    middle_frame, text="**** GAME LOG ****", font="Helvetica 13 bold",
foreground="blue").pack()
lbl_line = tk.Label(
    middle_frame,
text="************************************************************").pack()
round_frame = tk.Frame(middle_frame)
lbl_round = tk.Label(round_frame, text="Round")
lbl_round.pack()
lbl_your_choice = tk.Label(
    round_frame, text="Your choice: " + "None", font="Helvetica 13 bold")
lbl_your_choice.pack()
lbl_opponent_choice = tk.Label(round_frame, text="Opponent choice: " + "None")
lbl_opponent_choice.pack()
lbl_result = tk.Label(
    round_frame, text=" ", foreground="blue", font="Helvetica 14 bold")
lbl_result.pack()
round_frame.pack(side=tk.TOP)
final_frame = tk.Frame(middle_frame)
lbl_line = tk.Label(
    final_frame,
```

```
text="************************************************************").pack()
lbl_final_result = tk.Label(
    final_frame, text=" ", font="Helvetica 13 bold", foreground="blue")
lbl_final_result.pack()
lbl_line = tk.Label(
    final_frame,
text="************************************************************").pack()
final_frame.pack(side=tk.TOP)
middle_frame.pack_forget()
button_frame = tk.Frame(window_main)
photo_rock = PhotoImage(file=r"rock.gif")
photo_paper = PhotoImage(file=r"paper.gif")
photo_scissors = PhotoImage(file=r"scissors.gif")
btn_rock = tk.Button(
    button_frame,
    text="Rock",
    command=lambda: choice("rock"),
    state=tk.DISABLED,
    image=photo_rock,)
btn_paper = tk.Button(
    button_frame,
    text="Paper",
    command=lambda: choice("paper"),
    state=tk.DISABLED,
    image=photo_paper,)
btn_scissors = tk.Button(
    button_frame,
    text="Scissors",
    command=lambda: choice("scissors"),
    state=tk.DISABLED,
    image=photo_scissors,)
btn_rock.grid(row=0, column=0)
btn_paper.grid(row=0, column=1)
btn_scissors.grid(row=0, column=2)
button_frame.pack(side=tk.BOTTOM)
```

```
def game_logic(you, opponent):
    winner = ""
    rock = "rock"
    paper = "paper"
    scissors = "scissors"
    player0 = "you"
    player1 = "opponent"

    if you == opponent:
        winner = "draw"
    elif you == rock:
        if opponent == paper:
            winner = player1
        else:
            winner = player0
    elif you == scissors:
        if opponent == rock:
            winner = player1
        else:
            winner = player0
    elif you == paper:
        if opponent == scissors:
            winner = player1
        else:
            winner = player0
    return winner
def enable_disable_buttons(todo):
    if todo == "disable":
        btn_rock.config(state=tk.DISABLED)
        btn_paper.config(state=tk.DISABLED)
        btn_scissors.config(state=tk.DISABLED)
    else:
        btn_rock.config(state=tk.NORMAL)
        btn_paper.config(state=tk.NORMAL)
        btn_scissors.config(state=tk.NORMAL)
```

```
def connect():
    global your_name
    if len(ent_name.get()) < 1:
        tk.messagebox.showerror(
            title="ERROR!!!", message="You MUST enter your first name <e.g. John>)
    else:
        your_name = ent_name.get()
        lbl_your_name["text"] = "Your name: " + your_name
        connect_to_server(your_name)
def count_down(my_timer, nothing):
    global game_round
    if game_round <= TOTAL_NO_OF_ROUNDS:
        game_round = game_round + 1
    lbl_game_round["text"] = "Game round " + str(game_round) + " starts in"
    while my_timer > 0:
        my_timer = my_timer - 1
        print("game timer is: " + str(my_timer))
        lbl_timer["text"] = my_timer
        sleep(1)
    enable_disable_buttons("enable")
    lbl_round["text"] = "Round - " + str(game_round)
    lbl_final_result["text"] = " "
def choice(arg):
    global your_choice, client, game_round
    your_choice = arg
    lbl_your_choice["text"] = "Your choice: " + your_choice
    if client:
        dataToSend = "Game_Round" + str(game_round) + your_choice
        client.send(dataToSend.encode())
        enable_disable_buttons("disable")
def connect_to_server(name):
    global client, HOST_PORT, HOST_ADDR, your_name
    try:
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect((HOST_ADDR, HOST_PORT))
```

```
        client.send(name.encode())  # Send name to server after connecting
        # disable widgets
        btn_connect.config(state=tk.DISABLED)
        ent_name.config(state=tk.DISABLED)
        lbl_name.config(state=tk.DISABLED)
        enable_disable_buttons("disable")
        # start a thread to keep receiving message from server
        # do not block the main thread :)
        threading._start_new_thread(receive_message_from_server, (client, "m"))
    except Exception as e:
        tk.messagebox.showerror(
            title="ERROR!!!",
            message="Cannot connect to host: "
            + HOST_ADDR
            + " on port: "
            + str(HOST_PORT)
            + " Server may be Unavailable. Try again later"  )
```