PES UNIVERSITY

EC Campus, Bengaluru

Department of Computer Science & Engineering

COMPUTER NETWORKS - UE21CS252B

III Semester – I Section

ASSIGNMENT – 1

Online Multiplayer Game (Rock Paper Scissors Game)

Submitted to:

Dr. C.P Chavan

Submitted By:

Name: Somanshu B SRN: PES2UG21CS534

Name: Supratik Kar SRN: PES2UG21CS556

Name: Yash Kumar SRN: PES2UG21CS930

Table of Contents

Sl.No	**Title**	**Page No**
1	Abstract and scope of the Project	3
2	Project Description	4
3	Software Requirements	5
4	Source Code	6
5	Sample Output	11
6	Conclusion	13
7	References	14

Abstract and Scope of the Project

The Multiplayer Rock Paper Scissors game using TCP is a project that aims to demonstrate the implementation of a multiplayer game using socket programming and the Transmission Control Protocol (TCP) as the communication protocol. The game allows multiple players to connect to a central server and compete against each other in the classic **Rock Paper Scissors** game.

The project involves the development of both **server**-side and **client**-side applications using socket programming in Python. The server-side application creates a TCP socket and listens for incoming connections from clients. Once connected, the server manages the game logic, maintains game state, and facilitates communication between players. The client-side application connects to the server, sends player moves, receives game updates, and displays the game state to the user.

The game incorporates various networking concepts such as connection establishment, data exchange, error handling, and synchronization between multiple clients. It also utilizes the Rock Paper Scissors game logic, where players choose one of the three options (rock, paper, or scissors) and the winner is determined based on the predefined rules.

The project demonstrates the use of TCP as a reliable communication protocol for multiplayer gaming, ensuring that data is transmitted accurately and in the correct order. It also highlights the challenges and considerations in designing a multiplayer game, such as handling multiple connections, managing game state, and synchronizing game updates.

The outcome of the project is a fully functional Multiplayer Rock Paper Scissors game using TCP, providing an engaging and interactive gaming experience for players to compete against each other over a network. The project serves as a valuable example of socket programming and TCP implementation in the context of multiplayer gaming, with potential applications in various multiplayer game development scenarios. Further enhancements, such as adding additional features, improving performance, and expanding the player base, can be explored to extend the capabilities of the game. Overall, the Multiplayer Rock Paper Scissors game using TCP is a compelling demonstration of the power and versatility of socket programming and TCP in the realm of multiplayer gaming.

Project Description

(Detailed explanation about project implementation and details of functions defined)

- ***CLIENT.PY***

In client.py we have included the library pygame which helps us in making the game, moreover we have made buttons and made some adjustments to the window size and the buttons using python. Also, there is a message that pops up every time a round ends displaying “you’ve won” or “you’ve lost”.

- ***NETWORK.PY:***

In network.py we have made a connect function which connects the client to the server and we have also made the send function which send packets in the form of strings through a TCP stream. We connect the client and server by keep similar machine ip address in the server.py and client.py.

- ***SERVER.PY:***

In server.py we try to bind the client’s machine IP and port number to the server’s machine ip and port number. If there is any error in binding it throws an error. Here we also create threads and also provide a message if the server and client are connected, if they are connected the message says “connection established” if not then it says “connection not established”. If a user exits the game, then there is message that says “lost connection”.

- ***GAME.PY:***

In game.py we have written the logic as to which user wins, loses or ties while throwing either rock, paper or scissors.

Software Requirements

(Description about Programming Languages, APIs, Methods, etc.,)

- Python programming language and its various libraries are used for this project.

- Socket library:

  socket.socket(): This method creates a new socket object which is used to establish connections.

- socket.bind(host, port) :
  Binds the IP address and a port to the socket object so that any communication through a particular IP address and port is received by the socket object.

- socket.listen() :

  The socket object actively checks for any incoming connection requests.

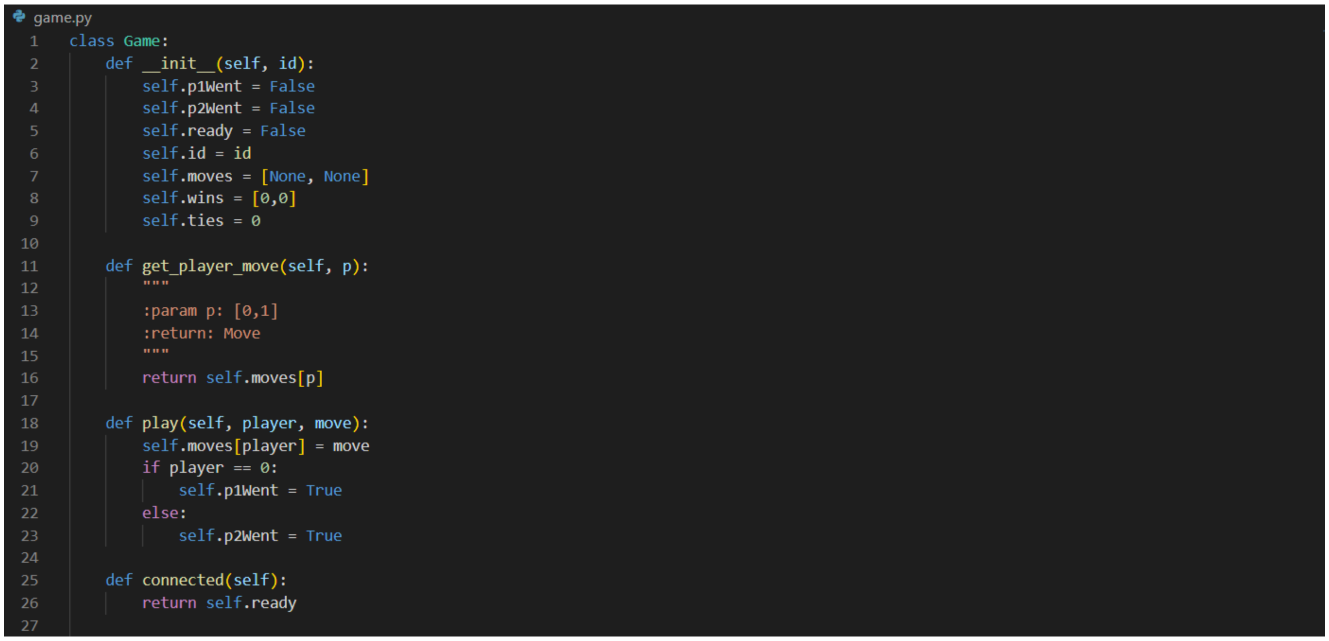
- socket.recv() :
  Receives the message transmitted by a client after the establishment of a connection.

- socket.sendall() :
  Sends user defined or program defined messages to the clients .

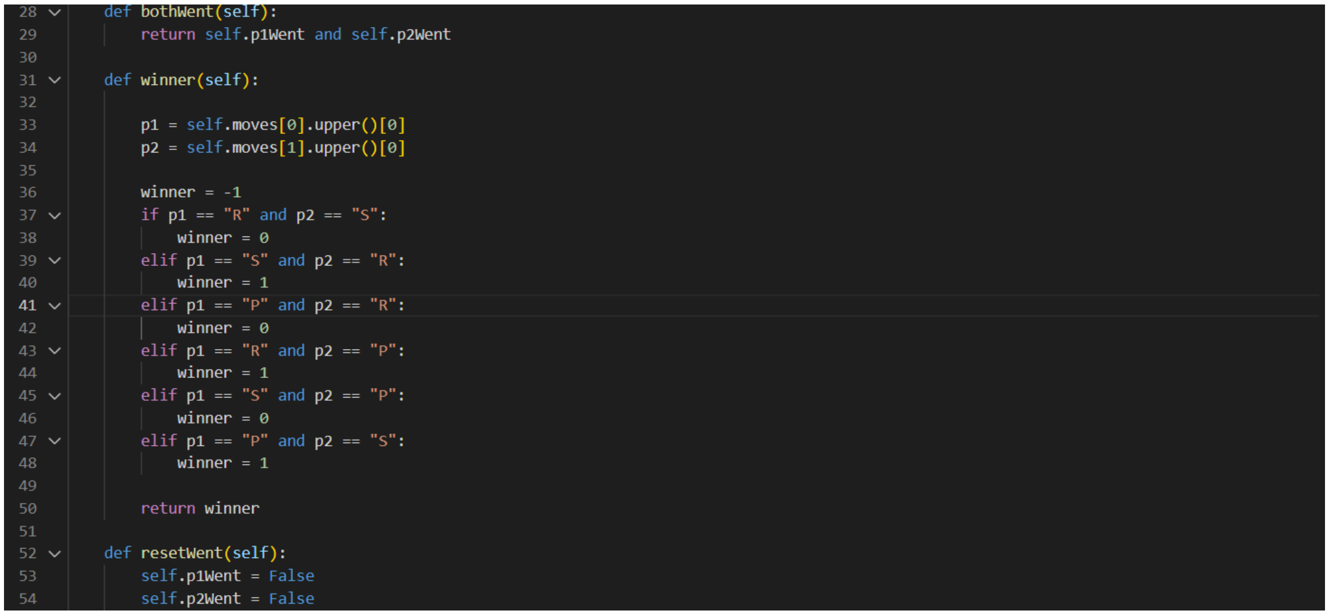
- Thread library:
  threading.Thread() :
  Creates a new process thread that can perform a different set of task independently.

Source Code

i. game.py

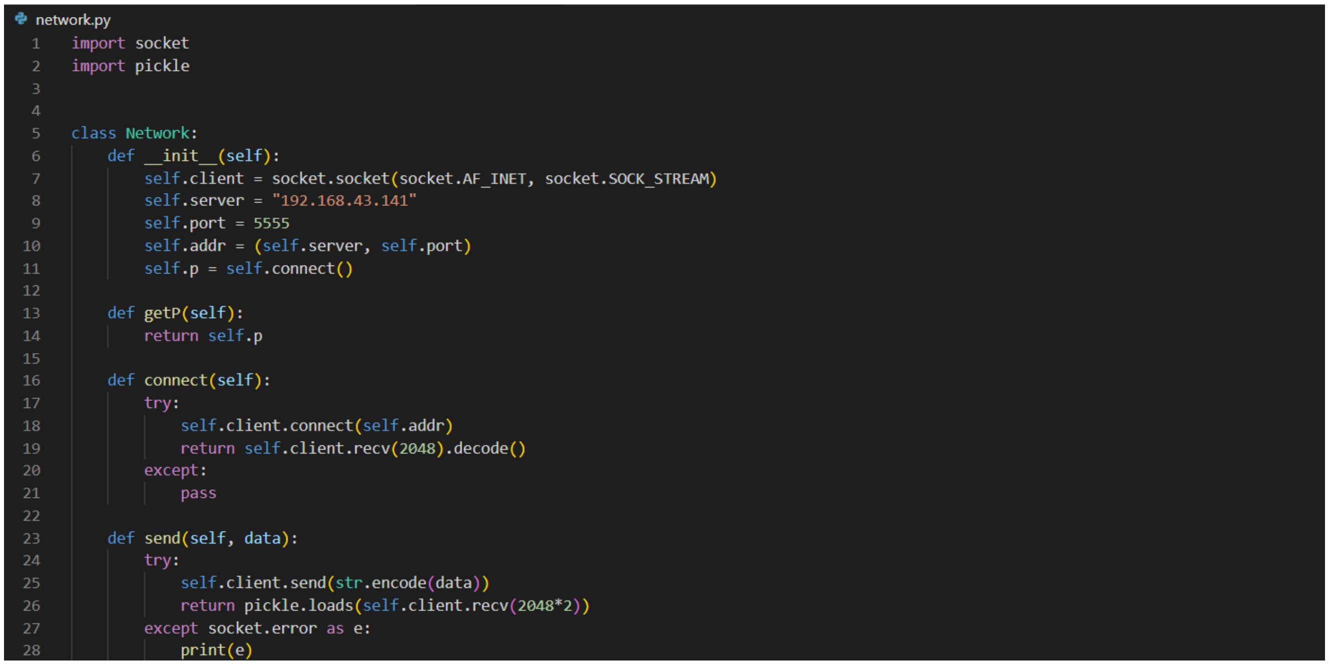


```python
class Game:
    def __init__(self, id):
        self.p1Went = False
        self.p2Went = False
        self.ready = False
        self.id = id
        self.moves = [None, None]
        self.wins = [0,0]
        self.ties = 0

    def get_player_move(self, p):
        """
        :param p: [0,1]
        :return: Move
        """
        return self.moves[p]

    def play(self, player, move):
        self.moves[player] = move
        if player == 0:
            self.p1Went = True
        else:
            self.p2Went = True

    def connected(self):
        return self.ready

```



```python
    def bothWent(self):
        return self.p1Went and self.p2Went

    def winner(self):

        p1 = self.moves[0].upper()[0]
        p2 = self.moves[1].upper()[0]

        winner = -1
        if p1 == "R" and p2 == "S":
            winner = 0
        elif p1 == "S" and p2 == "R":
            winner = 1
        elif p1 == "P" and p2 == "R":
            winner = 0
        elif p1 == "R" and p2 == "P":
            winner = 1
        elif p1 == "S" and p2 == "P":
            winner = 0
        elif p1 == "P" and p2 == "S":
            winner = 1

        return winner

    def resetWent(self):
        self.p1Went = False
        self.p2Went = False
```

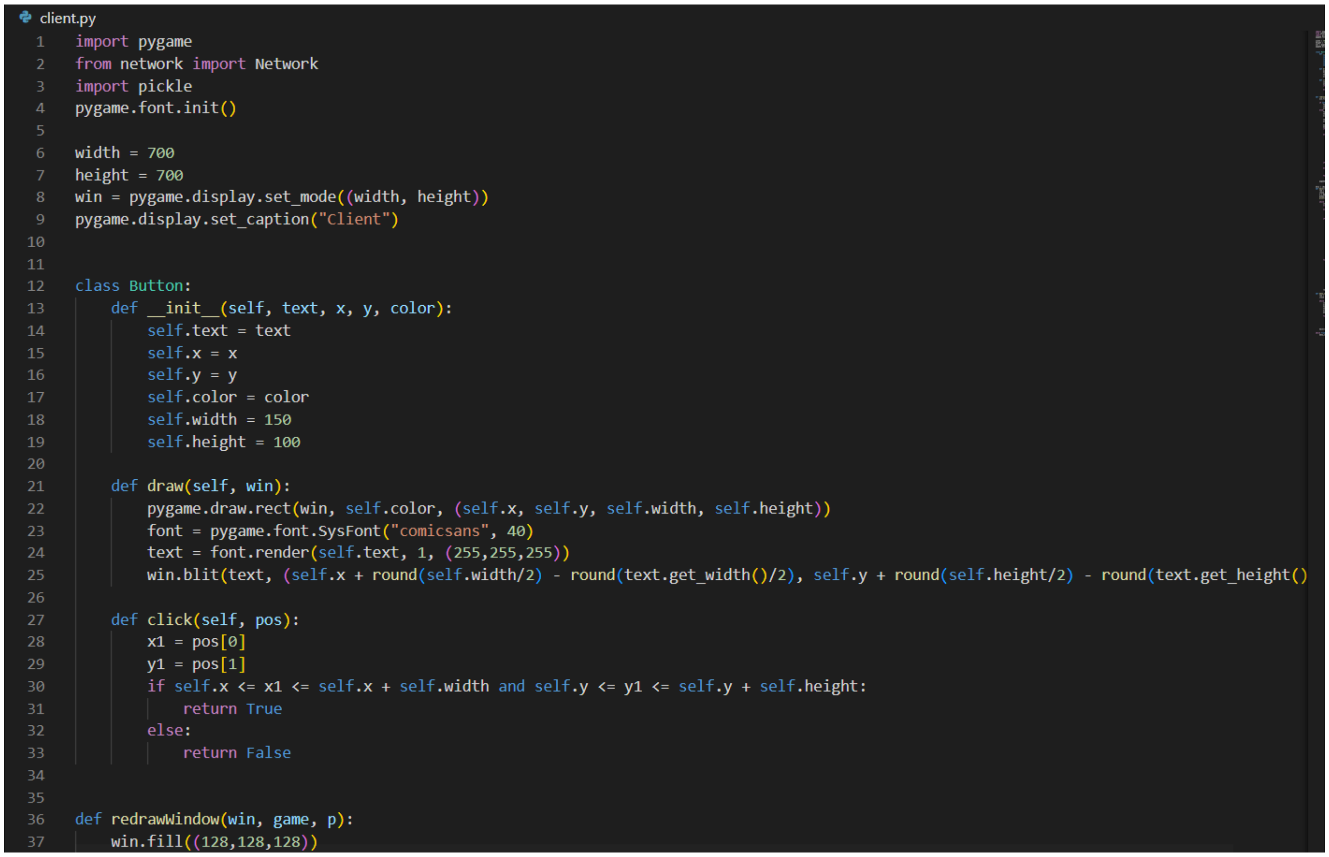
ii. Network.py



network.py

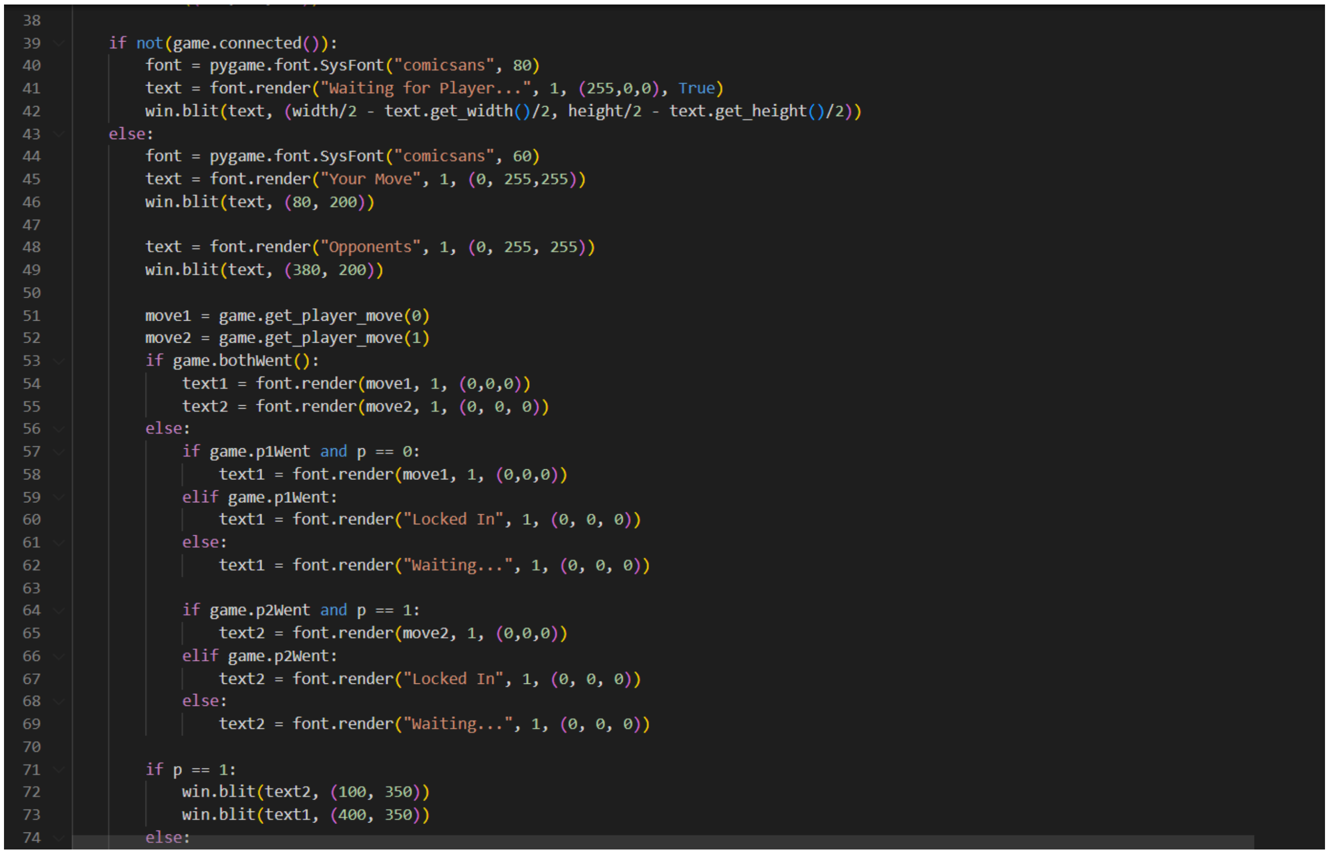
```python
import socket
import pickle


class Network:
    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "192.168.43.141"
        self.port = 5555
        self.addr = (self.server, self.port)
        self.p = self.connect()

    def getP(self):
        return self.p

    def connect(self):
        try:
            self.client.connect(self.addr)
            return self.client.recv(2048).decode()
        except:
            pass

    def send(self, data):
        try:
            self.client.send(str.encode(data))
            return pickle.loads(self.client.recv(2048*2))
        except socket.error as e:
            print(e)
```

iii. Client.py

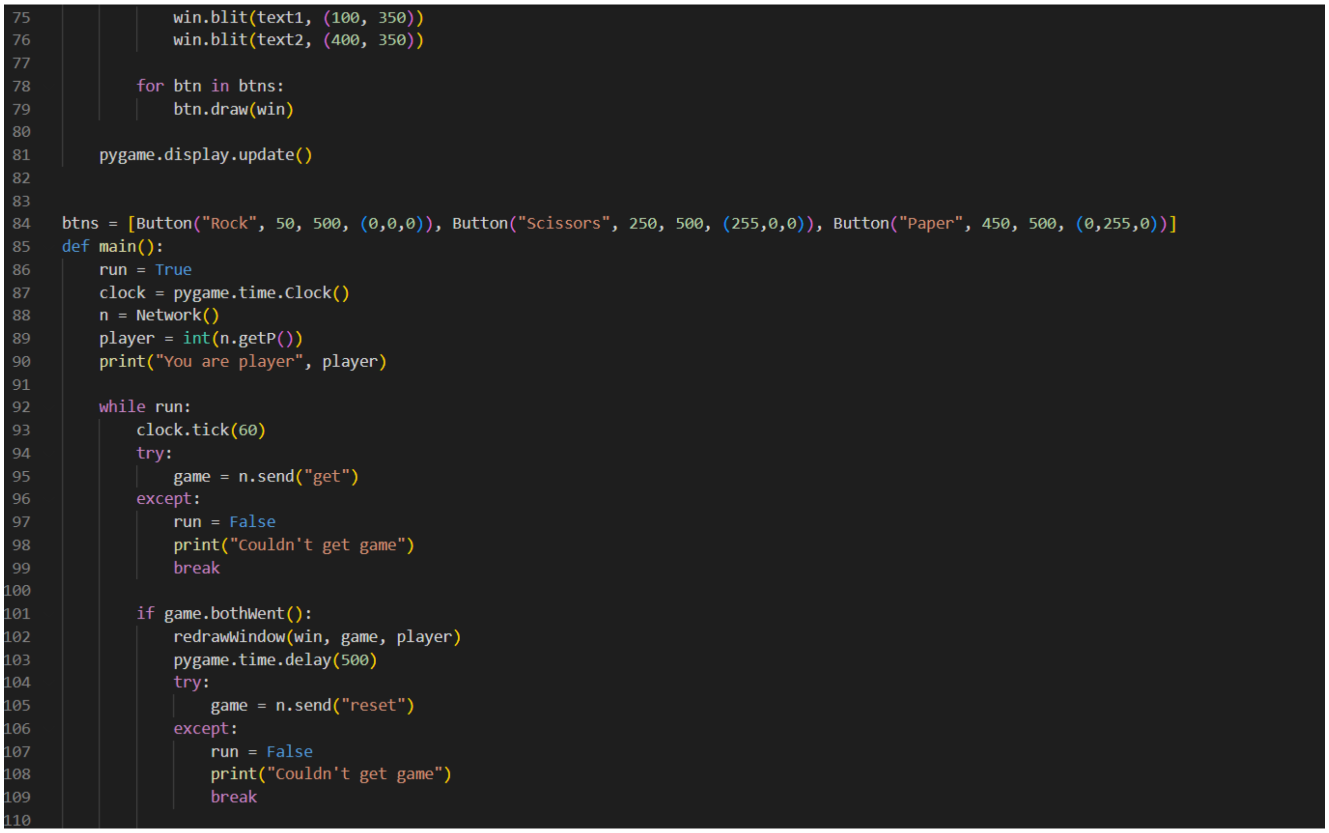


client.py

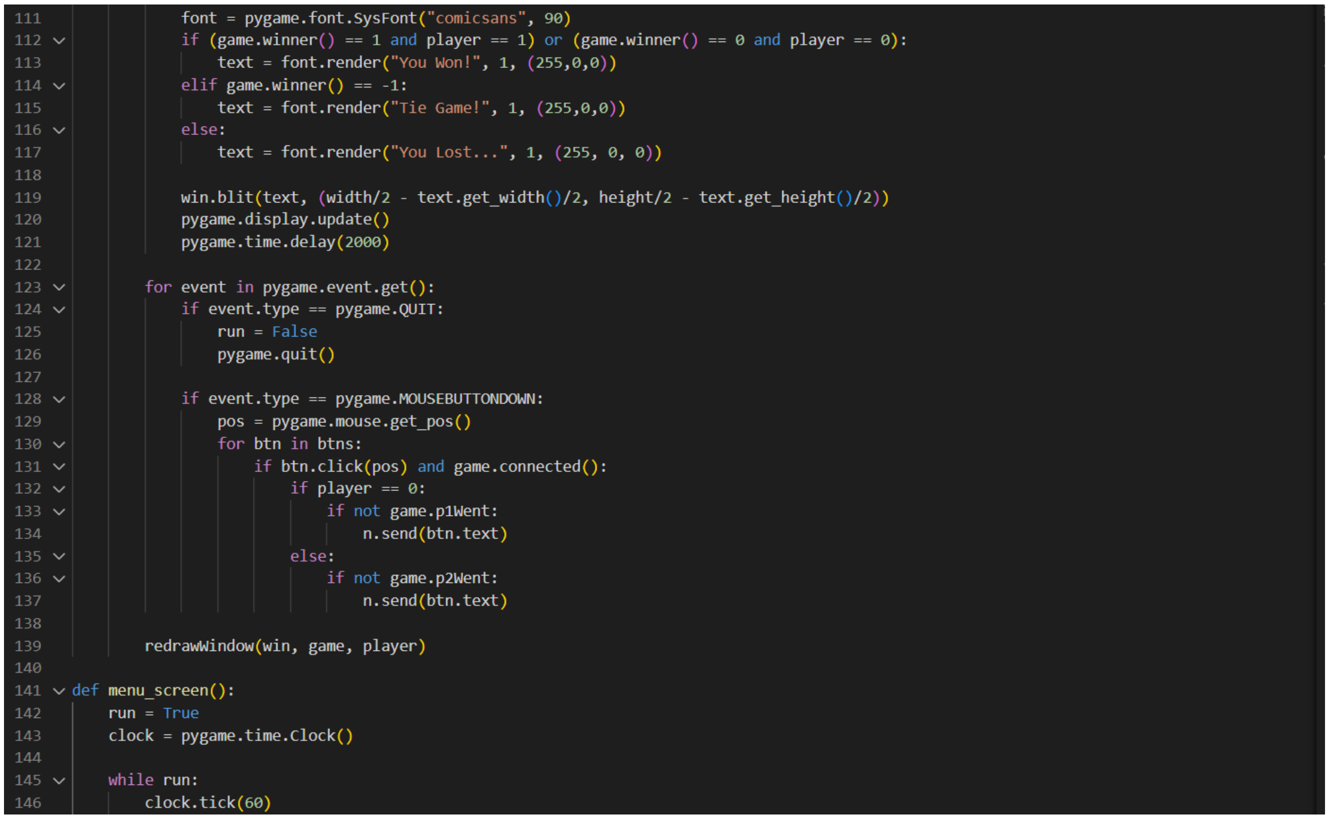
```python
import pygame
from network import Network
import pickle
pygame.font.init()

width = 700
height = 700
win = pygame.display.set_mode((width, height))
pygame.display.set_caption("Client")


class Button:
    def __init__(self, text, x, y, color):
        self.text = text
        self.x = x
        self.y = y
        self.color = color
        self.width = 150
        self.height = 100

    def draw(self, win):
        pygame.draw.rect(win, self.color, (self.x, self.y, self.width, self.height))
        font = pygame.font.SysFont("comicsans", 40)
        text = font.render(self.text, 1, (255,255,255))
        win.blit(text, (self.x + round(self.width/2) - round(text.get_width()/2), self.y + round(self.height/2) - round(text.get_height()

    def click(self, pos):
        x1 = pos[0]
        y1 = pos[1]
        if self.x <= x1 <= self.x + self.width and self.y <= y1 <= self.y + self.height:
            return True
        else:
            return False


def redrawWindow(win, game, p):
    win.fill((128,128,128))
```



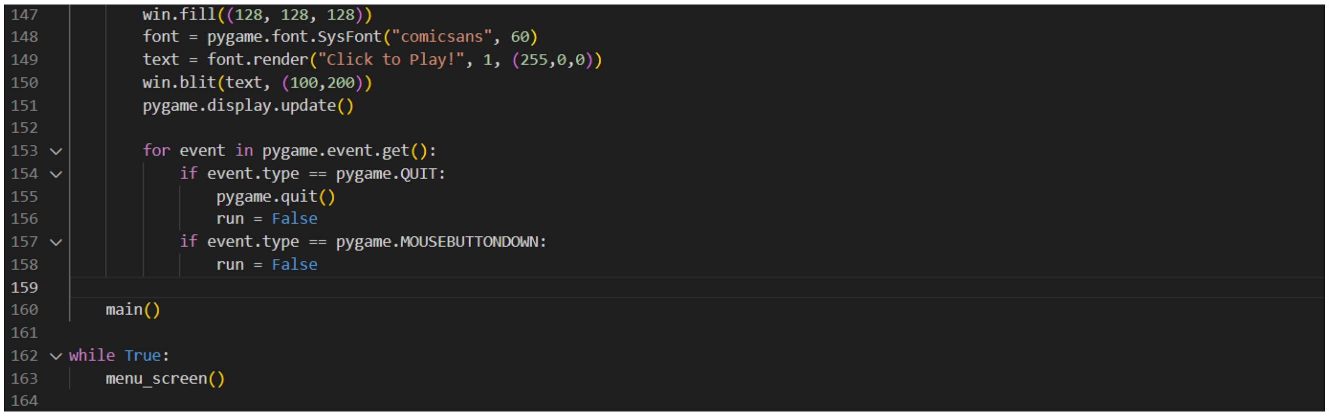
```python

    if not(game.connected()):
        font = pygame.font.SysFont("comicsans", 80)
        text = font.render("Waiting for Player...", 1, (255,0,0), True)
        win.blit(text, (width/2 - text.get_width()/2, height/2 - text.get_height()/2))
    else:
        font = pygame.font.SysFont("comicsans", 60)
        text = font.render("Your Move", 1, (0, 255,255))
        win.blit(text, (80, 200))

        text = font.render("Opponents", 1, (0, 255, 255))
        win.blit(text, (380, 200))

        move1 = game.get_player_move(0)
        move2 = game.get_player_move(1)
        if game.bothWent():
            text1 = font.render(move1, 1, (0,0,0))
            text2 = font.render(move2, 1, (0, 0, 0))
        else:
            if game.p1Went and p == 0:
                text1 = font.render(move1, 1, (0,0,0))
            elif game.p1Went:
                text1 = font.render("Locked In", 1, (0, 0, 0))
            else:
                text1 = font.render("Waiting...", 1, (0, 0, 0))

            if game.p2Went and p == 1:
                text2 = font.render(move2, 1, (0,0,0))
            elif game.p2Went:
                text2 = font.render("Locked In", 1, (0, 0, 0))
            else:
                text2 = font.render("Waiting...", 1, (0, 0, 0))

        if p == 1:
            win.blit(text2, (100, 350))
            win.blit(text1, (400, 350))
        else:
```



```python
            win.blit(text1, (100, 350))
            win.blit(text2, (400, 350))

        for btn in btns:
            btn.draw(win)

    pygame.display.update()


btns = [Button("Rock", 50, 500, (0,0,0)), Button("Scissors", 250, 500, (255,0,0)), Button("Paper", 450, 500, (0,255,0))]
def main():
    run = True
    clock = pygame.time.Clock()
    n = Network()
    player = int(n.getP())
    print("You are player", player)

    while run:
        clock.tick(60)
        try:
            game = n.send("get")
        except:
            run = False
            print("Couldn't get game")
            break

        if game.bothWent():
            redrawWindow(win, game, player)
            pygame.time.delay(500)
            try:
                game = n.send("reset")
            except:
                run = False
                print("Couldn't get game")
                break

```

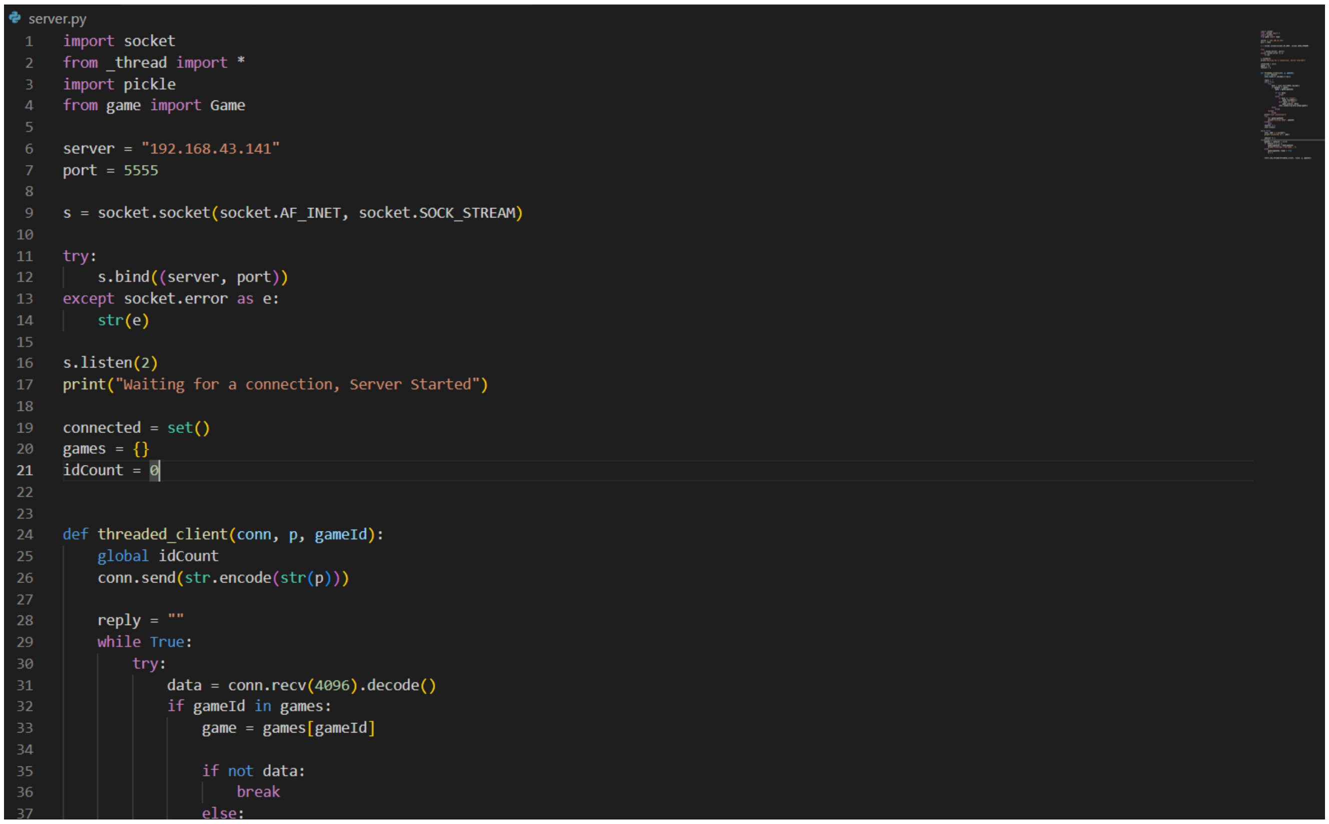


```python
            font = pygame.font.SysFont("comicsans", 90)
            if (game.winner() == 1 and player == 1) or (game.winner() == 0 and player == 0):
                text = font.render("You Won!", 1, (255,0,0))
            elif game.winner() == -1:
                text = font.render("Tie Game!", 1, (255,0,0))
            else:
                text = font.render("You Lost...", 1, (255, 0, 0))

            win.blit(text, (width/2 - text.get_width()/2, height/2 - text.get_height()/2))
            pygame.display.update()
            pygame.time.delay(2000)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                pos = pygame.mouse.get_pos()
                for btn in btns:
                    if btn.click(pos) and game.connected():
                        if player == 0:
                            if not game.p1Went:
                                n.send(btn.text)
                        else:
                            if not game.p2Went:
                                n.send(btn.text)

        redrawWindow(win, game, player)

def menu_screen():
    run = True
    clock = pygame.time.Clock()

    while run:
        clock.tick(60)
```



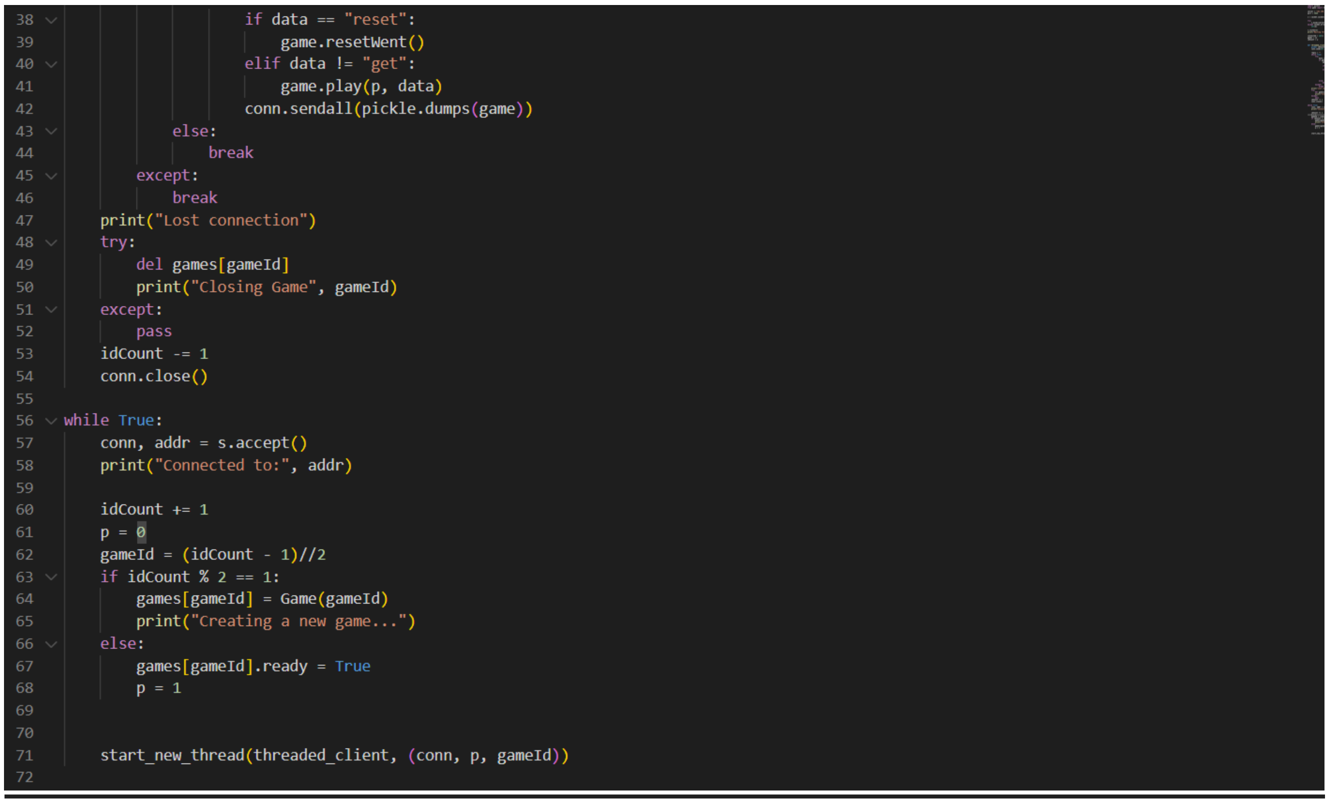
```python
        win.fill((128, 128, 128))
        font = pygame.font.SysFont("comicsans", 60)
        text = font.render("Click to Play!", 1, (255,0,0))
        win.blit(text, (100,200))
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                run = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                run = False

    main()

while True:
    menu_screen()

```

iv. Server.py



server.py

```python
import socket
from _thread import *
import pickle
from game import Game

server = "192.168.43.141"
port = 5555

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.bind((server, port))
except socket.error as e:
    str(e)

s.listen(2)
print("Waiting for a connection, Server Started")

connected = set()
games = {}
idCount = 0


def threaded_client(conn, p, gameId):
    global idCount
    conn.send(str.encode(str(p)))

    reply = ""
    while True:
        try:
            data = conn.recv(4096).decode()
            if gameId in games:
                game = games[gameId]

                if not data:
                    break
                else:
```



```python
                    if data == "reset":
                        game.resetWent()
                    elif data != "get":
                        game.play(p, data)
                    conn.sendall(pickle.dumps(game))
            else:
                break
        except:
            break
    print("Lost connection")
    try:
        del games[gameId]
        print("Closing Game", gameId)
    except:
        pass
    idCount -= 1
    conn.close()

while True:
    conn, addr = s.accept()
    print("Connected to:", addr)

    idCount += 1
    p = 0
    gameId = (idCount - 1)//2
    if idCount % 2 == 1:
        games[gameId] = Game(gameId)
        print("Creating a new game...")
    else:
        games[gameId].ready = True
        p = 1


    start_new_thread(threaded_client, (conn, p, gameId))

```

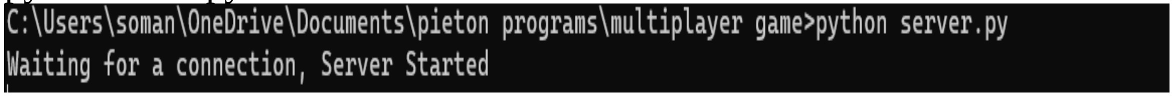
Sample Output

(Includes necessary output screenshots followed by a brief description)

Commands:

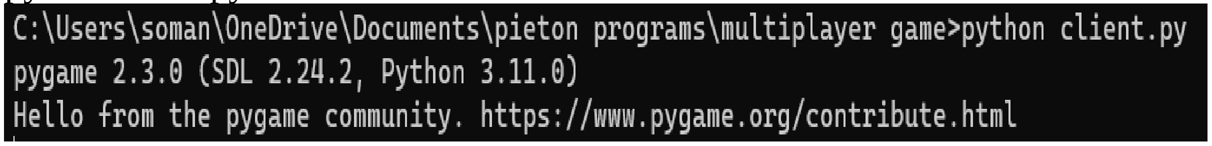
starting the server and client side using

- python sever.py



```
C:\Users\soman\OneDrive\Documents\pieton programs\multiplayer game>python server.py
Waiting for a connection, Server Started
```

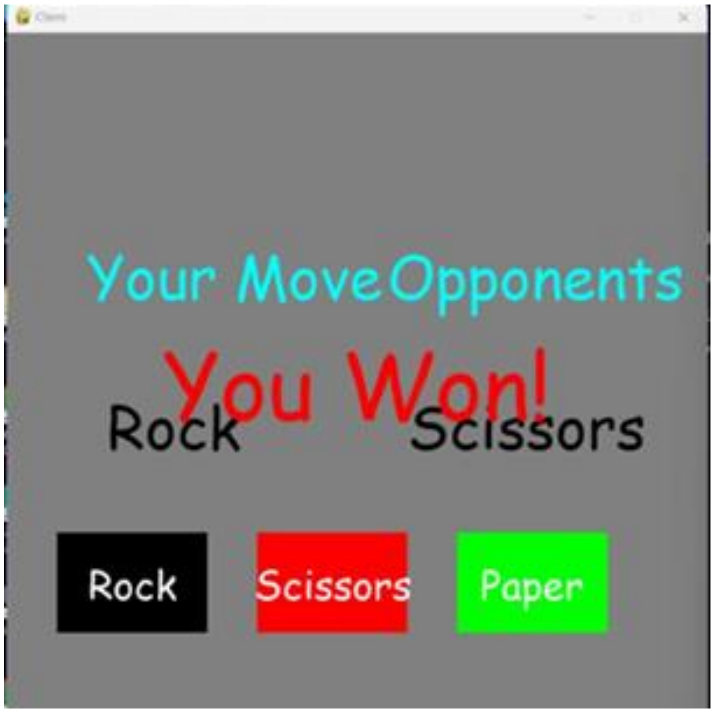
- python client.py

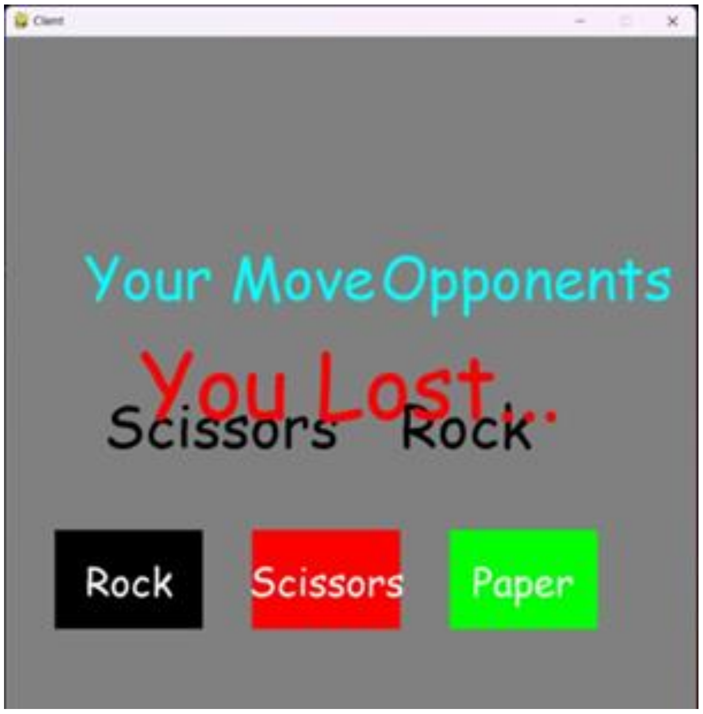


```
C:\Users\soman\OneDrive\Documents\pieton programs\multiplayer game>python client.py
pygame 2.3.0 (SDL 2.24.2, Python 3.11.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

Final Output Screenshot:

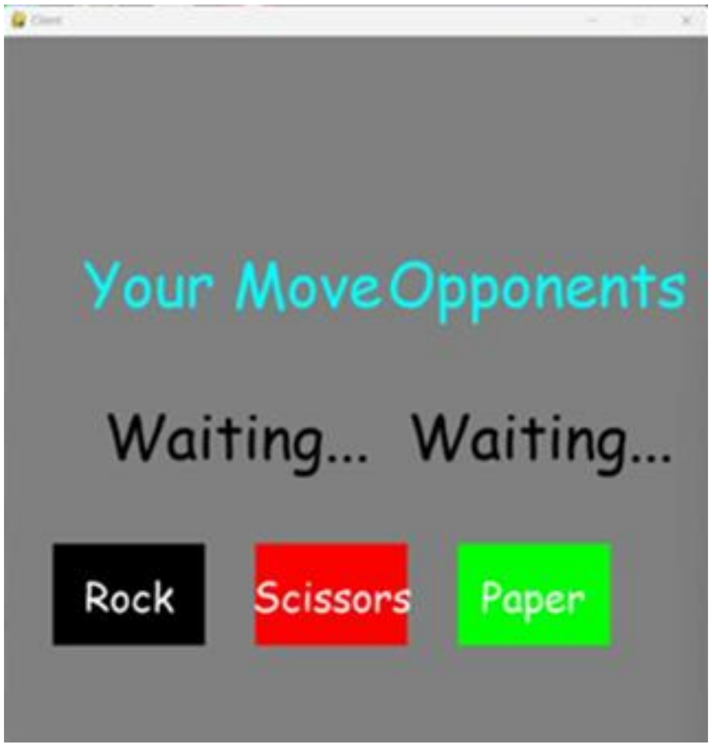
Shows the final Output as to which user "won" and which user "lost".

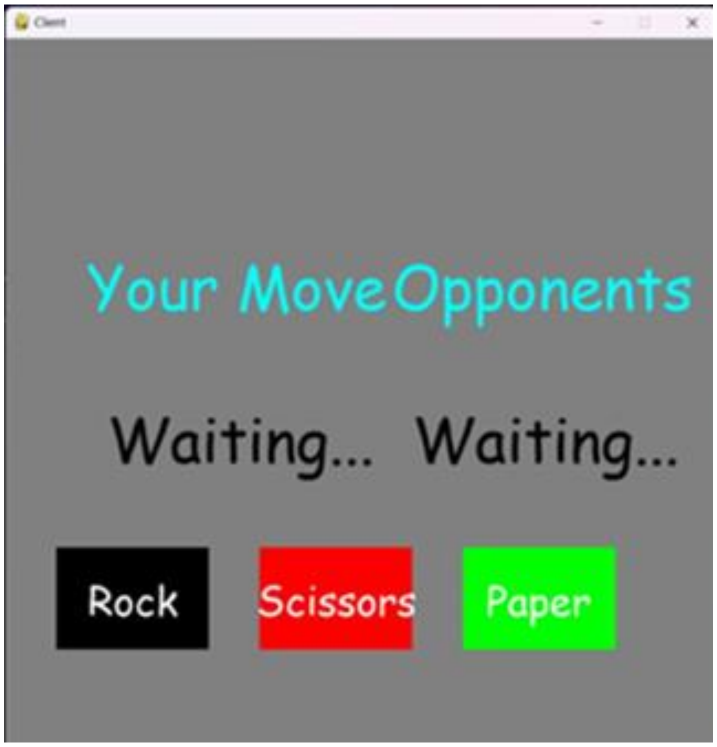




Waiting Output Screenshot:

This screenshot demonstrates that a user is wating for the **move of the opponent .**





Conclusion

In conclusion, our project on computer networks utilizing socket programming has been a significant endeavor that has provided us with a deep understanding of how data communication works at the network level. Our project has highlighted the importance of socket programming as a fundamental concept in computer networking, enabling the exchange of data over a network through sockets that serve as endpoints for communication. We have utilized various socket types, such as TCP, to implement different communication protocols based on the specific requirements of our project. We have also gained valuable insights into network architecture, including client-server models.

Further enhancements, such as adding additional features, improving performance, and expanding the player base, can be explored to extend the capabilities of the game. Overall, the Multiplayer Rock Paper Scissors game using TCP is a compelling demonstration of the power and versatility of socket programming and TCP in the realm of multiplayer gaming.

Overall, our project has provided us with invaluable practical experience in computer networking and socket programming, equipping us with the skills and knowledge to develop robust and efficient networked systems in the future.

References

- Python Documentation - https://docs.python.org/3/howto/sockets.html
- Tutorials Point- https://www.tutorialspoint.com/python/python_networking.html