COMPUTER COMMUNICATION NETWORKS

REPORT

TOPIC : 5*5 TIC-TAC-TOE GAME USING PYTHON

DEPARTMENT OF ECE

PES UNIVERSITY

BANGALORE

FACULTY INCHARGE :

PROF. PRAJEESHA

BY:

SHAKTHA SHETTY – PES1UG20EC336

YASHAS KUMARA ADI – PES1UG20EC349

SECTION – F

INDEX

THE PROBLEM STATEMENT:

To build a Tic-Tac-Toe gaming application using Socket Programming with Python.

INTRODUCTION:

A 5X5 Tic-Tac-Toe with socket programming between two different PC's. It takes maximum of 25 inputs and is a 2-player game. We have 1 server and 2 clients i.e, 2 players.

SOCKET PROGRAMMING:

It is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

A Socket has a typical flow of events. In a connection-oriented client-to-server model, the socket on the server process waits for request from a client. To do this, the server first establishes (binds) an address that client can use to find the server

Sockets are used to send messages across a network. They provide a form of inter-process communication (IPC).

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identity the application that data is destined to be sent to.

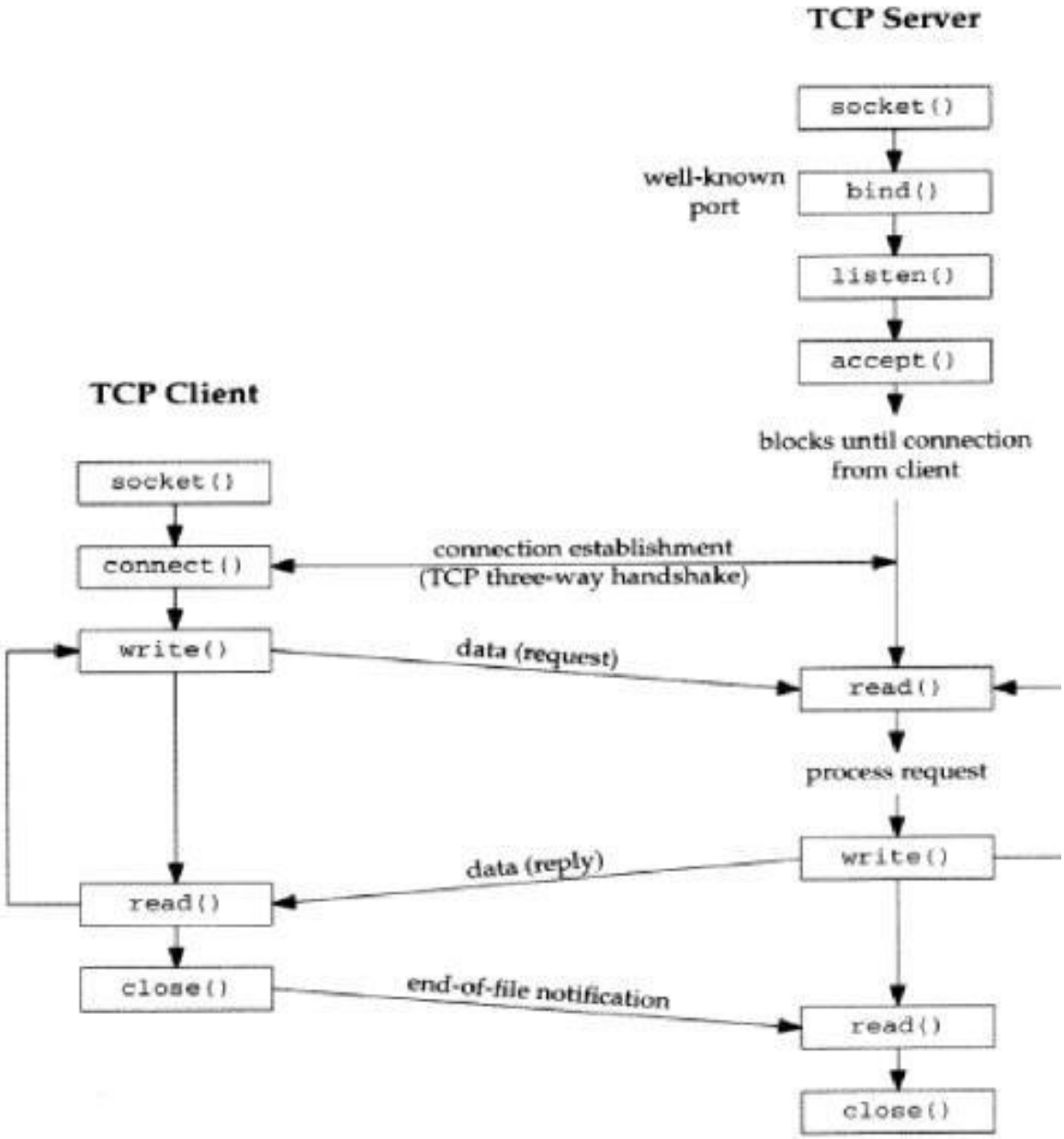
PYTHON :

Python is a High-level programming Language. It is a dynamic, free-open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. It is a developer-friendly language and can be easily understood.

MULTI-THREADING :

It is the ability of a processor to execute multiple threads concurrently. It rapidly switches the control of the CPU between threads.

FLOW CHART:


TCP Server
socket()
well-known port
bind()
listen()
accept()
blocks until connection from client
TCP Client
socket()
connect()
connection establishment (TCP three-way handshake)
write()
data (request)
read()
process request
write()
data (reply)
read()
close()
end-of-file notification
read()
close()

PROCEDURE:

1. This Tic-Tac-Toe game can be played between 2 players either on two different systems or a single system.
2. To start the game, first we run the server file on one of the systems. It binds to the port (default port given as 9999) and waits for the connection from clients/players.
3. Next step is to run the player file on the same system (as player 1). Once we run the player file, it asks us to input the IP address of the server system. After we enter the IP address, connection is established successfully between player 1 and the server.
4. Once this connection is established an interface pops up which is designed using pygame module in python. This interface is a 5*5 tic-tac-toe game and it says "waiting for peer" which means it is waiting for player 2 to join the server so that the game can commence.
5. Similarly, we run the player file on another system that is connected to the same network and we enter the IP address of the server so that the connection is successful and then the same interface pops up in this system too.
6. Now as both the players are connected to the server, the game can begin. First, player 1 gets a chance to choose any desired box and put a cross ('X') mark there. Following which, the player 2 gets a chance to enter a not ('O') mark in any of the remaining boxes.
7. The game continues in the same manner and the winner is declared when any one of the following conditions is satisfied-
    a. One among the two players manages to get 5 consecutive X's (for player 1) or 5 consecutive O's (for player 2) in a single row before the other does.
    b. One among the two players manages to get 5 consecutive X's (for player 1) or 5 consecutive O's (for player 2) in a single column before the other does.
    c. One among the two players manages to get 5 consecutive X's (for player 1) or 5 consecutive O's (for player 2) in a diagonal manner from top most box at the left to the bottom most box at the right or top most box at the right to bottom most box at the left.
8. When all the 25 boxes are filled and none of the above condition satisfies, a message pops up saying 'Draw Game' which indicates that there is no winner and the game is a draw.

CODE:

1. **Server.py**

```
import socket
import pickle
import time

s = socket.socket()
host = ""
port = 9999
matrix = [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],]
# print(matrix)

playerOne = 1
playerTwo = 2

playerConn = list()
playerAddr = list()

#server side validation is disabled to reduce latency
'''
def validate_input(x, y, conn):
    if x > 3 or y > 3:
        print("\nOut of bound! Enter again...\n")
        conn.send("Error".encode())
        return False
    elif matrix[x][y] != 0:
        print("\nAlready entered! Try again...\n")
        conn.send("Error".encode())
        return False
    return True
'''

def get_input(currentPlayer):
    if currentPlayer == playerOne:
        player = "Player One's Turn"
        conn = playerConn[0]
    else:
        player = "Player Two's Turn"
```

```
            conn = playerConn[1]
        print(player)
        send_common_msg(player)
        try:
            conn.send("Input".encode())
            data = conn.recv(2048 * 10)
            conn.settimeout(20)
            dataDecoded = data.decode().split(",")
            x = int(dataDecoded[0])
            y = int(dataDecoded[1])
            matrix[x][y] = currentPlayer
            send_common_msg("Matrix")
            send_common_msg(str(matrix))
        except:
            conn.send("Error".encode())
            print("Error occured! Try again..")

    def check_rows():
        # print("Checking rows")
        result = 0
        for i in range(5):
            if matrix[i][0] == matrix[i][1] and matrix[i][1] == matrix[i][2] and matrix[i][2]
    ==matrix[i][3] and matrix[i][3] == matrix[i][4]:
                result = matrix[i][0]
                if result != 0:
                    break
        return result

    def check_columns():
        # print("Checking cols")
        result = 0
        for i in range(5):
            if matrix[0][i] == matrix[1][i] and matrix[1][i] == matrix[2][i] and matrix[2][i] ==
    matrix[3][i] and matrix[3][i] == matrix[4][i]:
                result = matrix[0][i]
                if result != 0:
                    break
        return result

    def check_diagonals():
        # print("Checking diagonals")
        result = 0
        if matrix[0][0] == matrix[1][1] and matrix[1][1] == matrix[2][2] and matrix[2][2] ==
    matrix[3][3] and matrix[3][3] == matrix[4][4] :
```

```
        result = matrix[0][0]
    elif matrix[0][4] == matrix[1][3] and matrix[1][3] == matrix[2][2] and matrix[2][2]
== matrix[3][1] and matrix[4][0] == matrix[3][1]:
        result = matrix[0][4]
    return result

def check_winner():
    result = 0
    result = check_rows()
    if result == 0:
        result = check_columns()
    if result == 0:
        result = check_diagonals()
    return result

#Socket program
def start_server():
    #Binding to port 9999
    #Only two clients can connect
        s.bind((host, port))
        print("Tic Tac Toe server started \nBinding to port", port)
        s.listen(2)
        accept_players()
        if socket.error:
            print("Server binding error:", e)


#Accept player
#Send player number
def accept_players():
    try:
        for i in range(2):
            conn, addr = s.accept()
            msg = "<<< You are player {} >>>".format(i+1)
            conn.send(msg.encode())

            playerConn.append(conn)
            playerAddr.append(addr)
            print("Player {} - [{}:{}]".format(i+1, addr[0], str(addr[1])))

        start_game()
        s.close()
    except socket.error as e:
        print("Player connection error", e)
```

```
    except KeyboardInterrupt:
        print("\nKeyboard Interrupt")
        exit()
    except Exception as e:
      print("Error occurred:", e)

def start_game():
    result = 0
    i = 0
    while result == 0 and i < 25 :
      if (i%2 == 0):
        get_input(playerOne)
      else:
        get_input(playerTwo)
      result = check_winner()
      i = i + 1
      # print("Current count", i ,result == 0 and i < 9, "Result = ", result)

    send_common_msg("Over")

    if result == 1:
      lastmsg = "Player One is the winner!!"
    elif result == 2:
      lastmsg = "Player Two is the winner!!"
    else:
      lastmsg = "Draw game!! Try again later!"

    send_common_msg(lastmsg)
    time.sleep(10)
    for conn in playerConn:
      conn.close()


def send_common_msg(text):
    playerConn[0].send(text.encode())
    playerConn[1].send(text.encode())
    time.sleep(1)

start_server()
```

2. **Player.py**

```
import pygame
import socket
import time
import threading

s = socket.socket()
host = input("Enter the server IP:")
port = 9999

playerOne = 1
playerOneColor = (255, 0, 0)
playerTwo = 2
playerTwoColor = (0, 0, 255)
bottomMsg = ""
msg = "Waiting for peer"
currentPlayer = 0
xy = (-1, -1)
allow = 0 #allow handling mouse events
matrix = [[0, 0, 0, 0, 0 ], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0]]

#Create worker threads
def create_thread(target):
   t = threading.Thread(target = target) #argument - target function
   t.daemon = True
   t.start()

#initialize
pygame.init()

width = 750
height = 750
screen = pygame.display.set_mode((width, height))

#set title
pygame.display.set_caption("Tic Tac Toe")

#set icon
icon = pygame.image.load("tictactoe.png")
pygame.display.set_icon(icon)
```

```
#fonts
bigfont = pygame.font.Font('freesansbold.ttf', 64)
smallfont = pygame.font.Font('freesansbold.ttf', 32)
backgroundColor = (255, 255, 255)
titleColor = (0, 0, 0)
subtitleColor = (128, 0, 255)
lineColor = (0, 0, 0)

def buildScreen(bottomMsg, string, playerColor = subtitleColor):
    screen.fill(backgroundColor)
    if "One" in string or "1" in string:
        playerColor = playerOneColor
    elif "Two" in string or "2" in string:
        playerColor = playerTwoColor

    #vertical lines
    pygame.draw.line(screen, lineColor, (250-2, 150), (250-2, 650), 4)
    pygame.draw.line(screen, lineColor, (350-2, 150), (350-2, 650), 4)
    pygame.draw.line(screen, lineColor, (450-2, 150), (450-2, 650), 4)
    pygame.draw.line(screen, lineColor, (550-2, 150), (550-2, 650), 4)
    #horizontal lines
    pygame.draw.line(screen, lineColor, (150, 250-2), (650, 250-2), 4)
    pygame.draw.line(screen, lineColor, (150, 350-2), (650, 350-2), 4)
    pygame.draw.line(screen, lineColor, (150, 450-2), (650, 450-2), 4)
    pygame.draw.line(screen, lineColor, (150, 550-2), (650, 550-2), 4)
    title = bigfont.render("TIC TAC TOE", True, titleColor)
    screen.blit(title, (140, 0))
    subtitle = smallfont.render(str.upper(string), True, playerColor)
    screen.blit(subtitle, (200, 90))
    centerMessage(bottomMsg, playerColor)

def centerMessage(msg, color = titleColor):
    pos = (190, 700)
    # screen.fill(backgroundColor)
    if "One" in msg or "1" in msg:
        color = playerOneColor
    elif "Two" in msg or "2" in msg:
        color = playerTwoColor
    msgRendered = smallfont.render(msg, True, color)
    screen.blit(msgRendered, pos)

def printCurrent(current, pos, color):
    currentRendered = bigfont.render(str.upper(current), True, color)
```

```
    screen.blit(currentRendered, pos)

def printMatrix(matrix):
    for i in range(5):
        #When row increases, y changes
        y = int((i + 1.75) * 100)
        for j in range(5):
            #When col increases, x changes
            x =  int((j + 1.75) * 100)
            current = " "
            color = titleColor
            if matrix[i][j] == playerOne:
                current = "X"
                color = playerOneColor
            elif matrix[i][j] == playerTwo:
                current = "O"
                color = playerTwoColor
            printCurrent(current, (x, y), color)

def validate_input(x, y):
    if x > 5 or y > 5:
        print("\nOut of bound! Enter again...\n")
        return False
    elif matrix[x][y] != 0:
        print("\nAlready entered! Try again...\n")
        return False
    return True

def handleMouseEvent(pos):
    x = pos[0]
    y = pos[1]
    global currentPlayer
    global xy
    if(x < 150 or x > 650 or y < 150 or y > 650):
        xy = (-1, -1)
    else:
        # When x increases, column changes
        col = int(x/100 - 1.5)
        # When y increases, row changes
        row = int(y/100 - 1.5)
        print("({}, {})".format(row,col))
        if validate_input(row, col):
            matrix[row][col] = currentPlayer
            xy = (row,col)
```

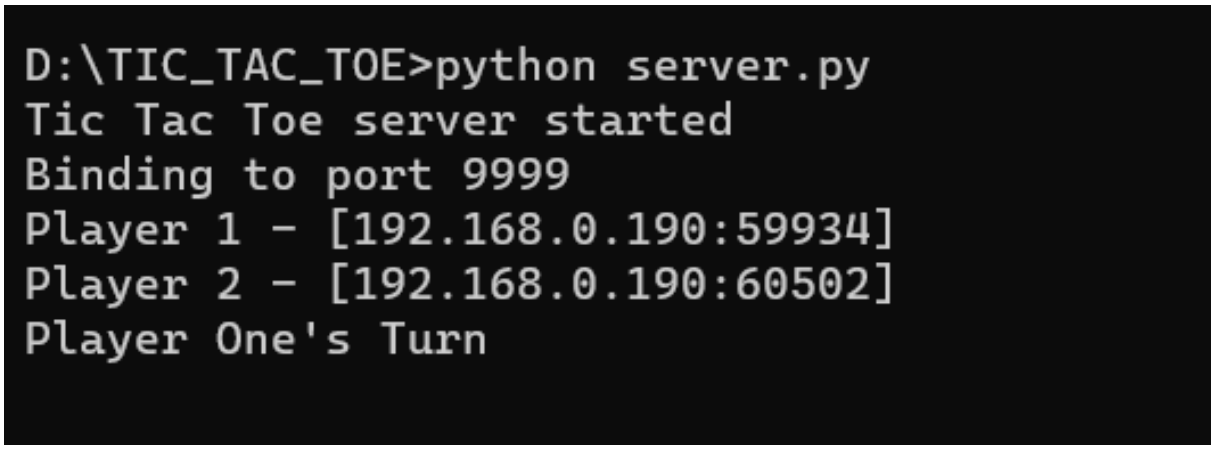
```
def start_player():
    global currentPlayer
    global bottomMsg
    try:
        s.connect((host, port))
        print("Connected to :", host, ":", port)
        recvData = s.recv(2048 * 10)
        bottomMsg = recvData.decode()
        if "1" in bottomMsg:
            currentPlayer = 1
        else:
            currentPlayer =2
        start_game()
        s.close()
    except socket.error as e:
        print("Socket connection error:", e)

def start_game():
    running = True
    global msg
    global matrix
    global bottomMsg
    create_thread(accept_msg)
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            if event.type == pygame.MOUSEBUTTONUP:
                pos = pygame.mouse.get_pos()
                if allow:
                    handleMouseEvent(pos)

        if msg == "":
            break

        buildScreen(bottomMsg, msg)
        printMatrix(matrix)
        pygame.display.update()

def accept_msg():
    global matrix
    global msg
    global bottomMsg
```

```
global allow
global xy
while True:
    try:
        recvData = s.recv(2048 * 10)
        recvDataDecode = recvData.decode()
        buildScreen(bottomMsg, recvDataDecode)

        if recvDataDecode == "Input":
            failed = 1
            allow = 1
            xy = (-1, -1)
            while failed:
                try:
                    if xy != (-1, -1):
                        coordinates = str(xy[0])+"," + str(xy[1])
                        s.send(coordinates.encode())
                        failed = 0
                        allow = 0
                except:
                    print("Error occured....Try again")

        elif recvDataDecode == "Error":
            print("Error occured! Try again..")

        elif recvDataDecode == "Matrix":
            print(recvDataDecode)
            matrixRecv = s.recv(2048 * 100)
            matrixRecvDecoded = matrixRecv.decode("utf-8")
            matrix = eval(matrixRecvDecoded)

        elif recvDataDecode == "Over":
            msgRecv = s.recv(2048 * 100)
            msgRecvDecoded = msgRecv.decode("utf-8")
            bottomMsg = msgRecvDecoded
            msg = "-_-Game Over-_-"
            break
        else:
            msg = recvDataDecode

    except KeyboardInterrupt:
        print("\nKeyboard Interrupt")
        time.sleep(1)
        break
```

```
        except:
            print("Error occured")
            break

    start_player()
```

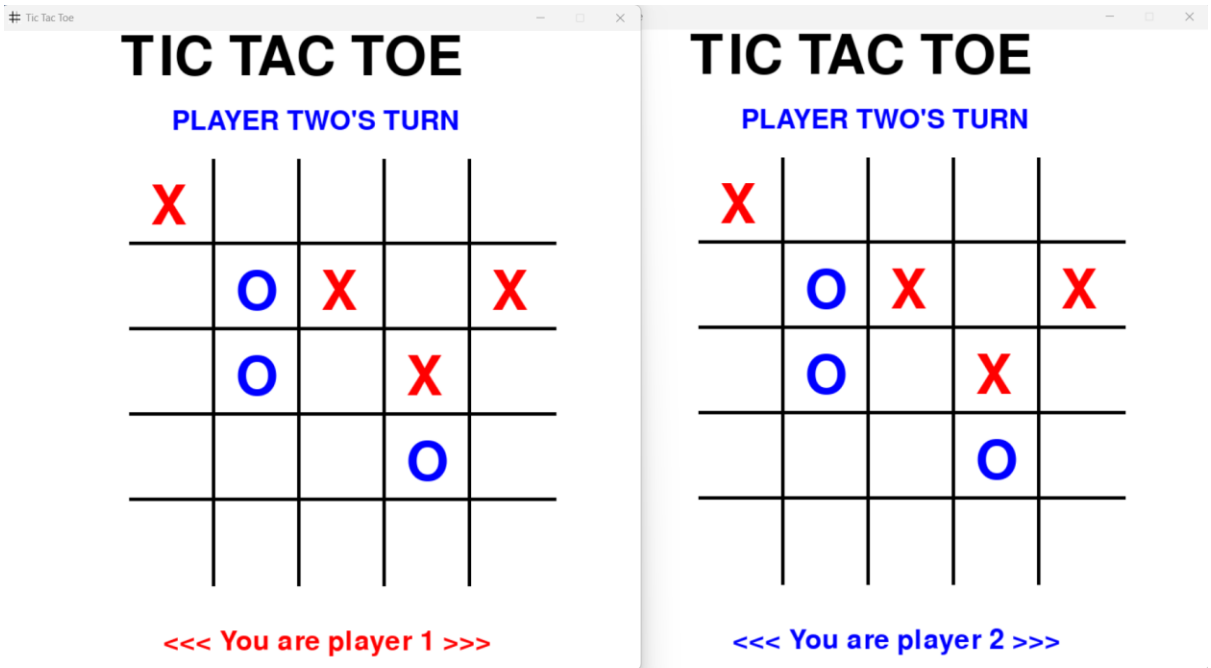
OUTPUT:

1. Running the server.py file

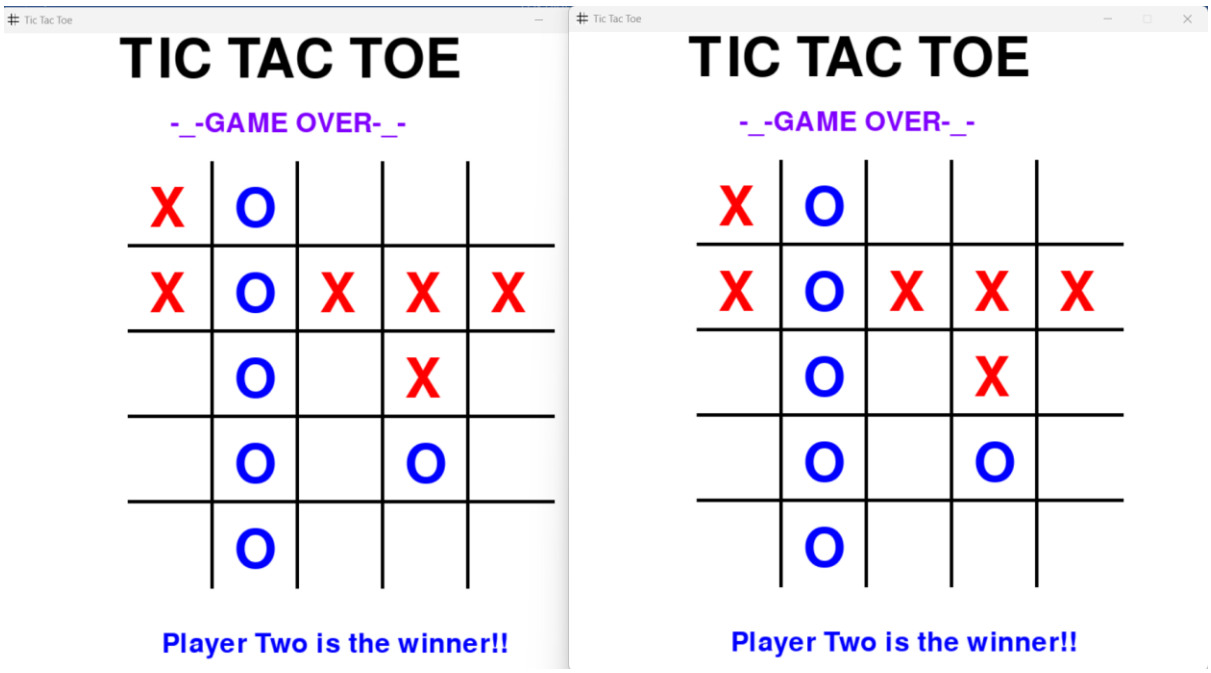


2. Connecting both the players and waiting for player 1 to play.

3. Both the players playing chance by chance.



4. Winner is declared.



CONCLUSION:

We built a 5*5 tic-tac-toe multiplayer game using socket programming in python. We used pygame module for designing the interface. There are 2 possible cases in this game, either the game will end being a draw or any one of the two players is declared as the winner based on the conditions listed above. Generally a tic-tac-toe game is played on a 3*3 matrix but to increase the complexity and to make it a bit more interesting game, we designed this game on a 5*5 matrix by modifying the logics of 3*3 into a 5*5 game. In real world this game can be played by any two people who are far away from each other and they don't need a same network to play this game. As this is a demonstration of the same we had to use same network for both the players to play the game.

REFERENCES:

1. https://www.geeksforgeeks.org/implementation-of-tic-tac-toe-game/
2. https://en.wikipedia.org/wiki/Tic-tac-toe

HYPERLINK TO THE RESOURCES:

https://github.com/Shakthashetty274/TIC-TAC-TOE