

**A Mini- Project Report**  
**on**  
**“Python Online Multiplayer: Rock paper scissors”**

Submitted to the  
Pune Institute of Computer  
Technology, Pune In partial fulfillment for  
the award of the Degree of Bachelor of  
Engineering  
in  
Information Technology  
by

Esha Anvekar	43205
Vaibhav Bichave	43209
Disha Chavan	43212
Megha Dandapat	43217

Under the guidance of

**Mr. Sachin Pande**



Department Of Information Technology  
Pune Institute of Computer Technology College of Engineering  
Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

**2022-2023**

# CERTIFICATE

This is to certify that the project report entitled

**Python Online Multiplayer: Rock paper  
scissors**

## Submitted by

Esha Anvekar	43205
Vaibhav Bichave	43209
Disha Chavan	43212
Megha Dandapat	43217

is a bonafide work carried out by them under the supervision of Mr. Sachin Pande and it is approved for the partial fulfillment of the requirement of **Lab Practice-V** for the award of the Degree of Bachelor of Engineering (Information Technology).

**Mr. Sachin Pande**

Lab Teacher

Department of Information Technology

**Dr. A. S. Gotkar**

Head of Department

Department of Information Technology

Place:

Date:

## II

### **ACKNOWLEDGEMENT**

We thank everyone who has helped and provided valuable suggestions for successfully creating a wonderful project.

We are very grateful to our guide, Mr. Sachin Pande, Head of Department Dr. A. S. Ghotkar and our principal Dr. S.T. Gandhe. They have been very supportive and have ensured that all facilities remained available for smooth progress of the project.

We would like to thank our professor and Mr. Sachin Pande for providing very valuable and timely suggestions and help.

Student Name  
Esha Anvekar  
Vaibhav Bichave  
Disha Chavan  
Megha Dandapat

### III

## **ABSTRACT**

The Rock Paper Scissors game developed using Python and Pygame is a two-player game that enables players to compete against each other using their network connection. The game is built on a client-server architecture, where players connect to the server via sockets to play the game.

The game uses a graphical interface developed using Pygame, which displays the game options and the score of each player. Players select their choices by clicking on the respective icon on the screen, and the game logic implemented in Python determines the winner based on the standard Rock Paper Scissors rules.

The networking aspect of the game is facilitated by the use of sockets, which enable players to communicate their choices to the server and receive the choices of their opponent. The server manages the game logic and sends updates to the players' screens regarding the current score and the winner of each round.

Overall, this implementation of Rock Paper Scissors provides an interactive and engaging experience for players, allowing them to test their skills against other players over a network connection.

## IV

### LIST OF FIGURES

Figure Number	Figure Title	Page Number
1	System architecture diagram	9
2	Project flow diagram	10
3	Code snippet	11
4	Code snippet	11
5	Output	12

## **LIST OF TOPICS**

1. INTRODUCTION
2. SCOPE AND OBJECTIVE
3. SYSTEM ARCHITECTURE/PROJECT FLOW
4. CODE AND SNAPSHOT
5. RESULT
6. CONCLUSION AND FUTURE SCOPE

# 1. INTRODUCTION

Video games that support simultaneous play between multiple players are known as multiplayer games. On the same device locally or through a network, like the internet, these games can be played.

Cooperative or competitive multiplayer games may incorporate a variety of gameplay aspects, including teamwork, strategy, communication, and skill. Players may be given the option to cooperate to perform tasks, compete with one another for points or other incentives, or combine all of these options.

Making multiplayer games requires the use of various technologies and tools, including programming languages, game engines, and networking libraries. Multiplayer games are frequently made using programming languages including C++, Java, Python, and JavaScript. The tools and functions required to develop networking and game logic are provided by these languages. A variety of tools are available for creating and managing game materials, physics, animations, and other features in game engines like Unity, Unreal Engine, and Godot. Additionally, these engines provide networking capabilities and other elements required for building multiplayer games. The ready-to-use networking capabilities offered by networking libraries like Photon, Mirror, and Netty can be included into game engines or game code. These libraries offer functions including server/client communication, game state synchronization, and network latency handling.

A selection of such game development tools are offered by the Python package Pygame. It is constructed on top of the Simple DirectMedia Layer (SDL) library, which gives users access to input devices, graphics, and audio across several platforms. Sprite management, collision detection, and event handling are just a few of the functions and classes offered by Pygame that make it simpler to develop 2D games.

Pygame enables programmers to construct games that work on a variety of operating systems, including Windows, Linux, and macOS. Additionally, it supports a number of input methods, such as touchscreens and game controllers. Pygame can handle 3D graphics using OpenGL and supports a large number of graphic formats, such as PNG, JPG, and GIF.

## **2. SCOPE AND OBJECTIVES**

- A clear understanding of the game mechanics, rules, and requirements, as well as the type of multiplayer experience.
- Choosing a networking model for the game, such as client-server or peer-to-peer considering factors such as latency, security, and scalability.
- Implementing the necessary networking functionality including features such as sending and receiving data between clients and servers, managing connections, and synchronizing game state across clients.
- Form a game logic that handles player input, updates the game state, and sends data to clients.
- Testing the game to ensure that it works as expected and is free from bugs and glitches.



### 3. SYSTEM ARCHITECTURE/PROJECT FLOW

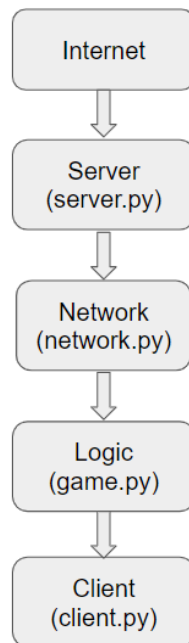


fig 1: System architecture diagram

The diagram in fig 1. shows the different components of the system and their relationships. The Internet provides the connection between the client and server. The server handles incoming connections from clients, manages game state, and communicates with clients. The network layer provides low-level networking functionality such as socket management and protocol definitions. The game logic layer contains the core logic of the game, such as defining the rules and determining the outcome of each round. The client handles user input, displays game results, and communicates with the server to send and receive game data.

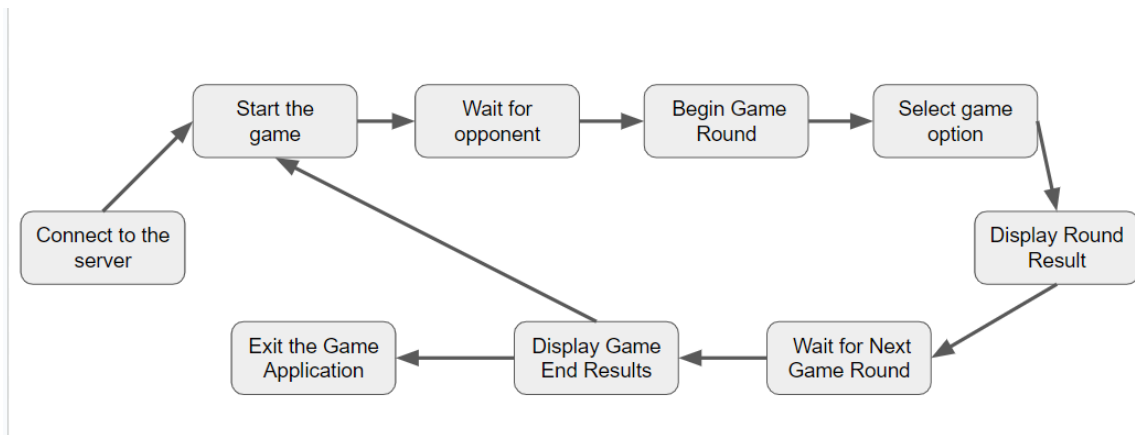
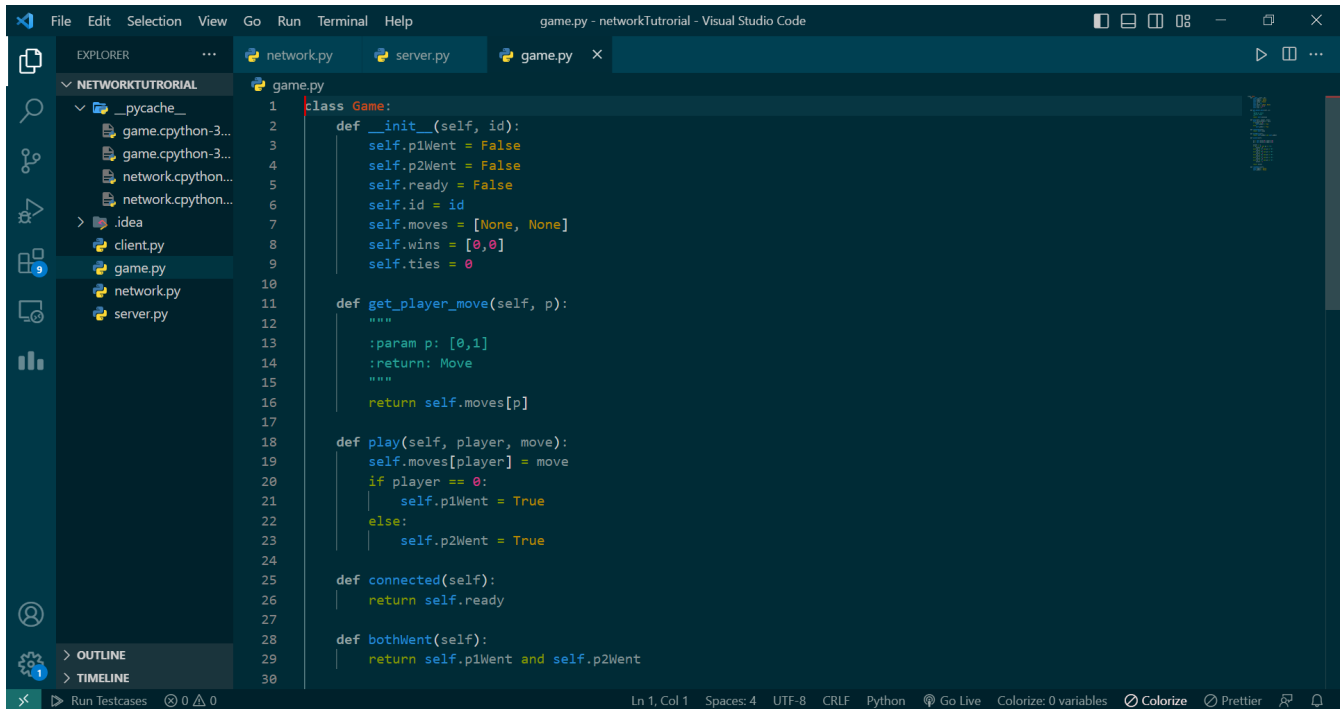


fig 2: Project flow diagram

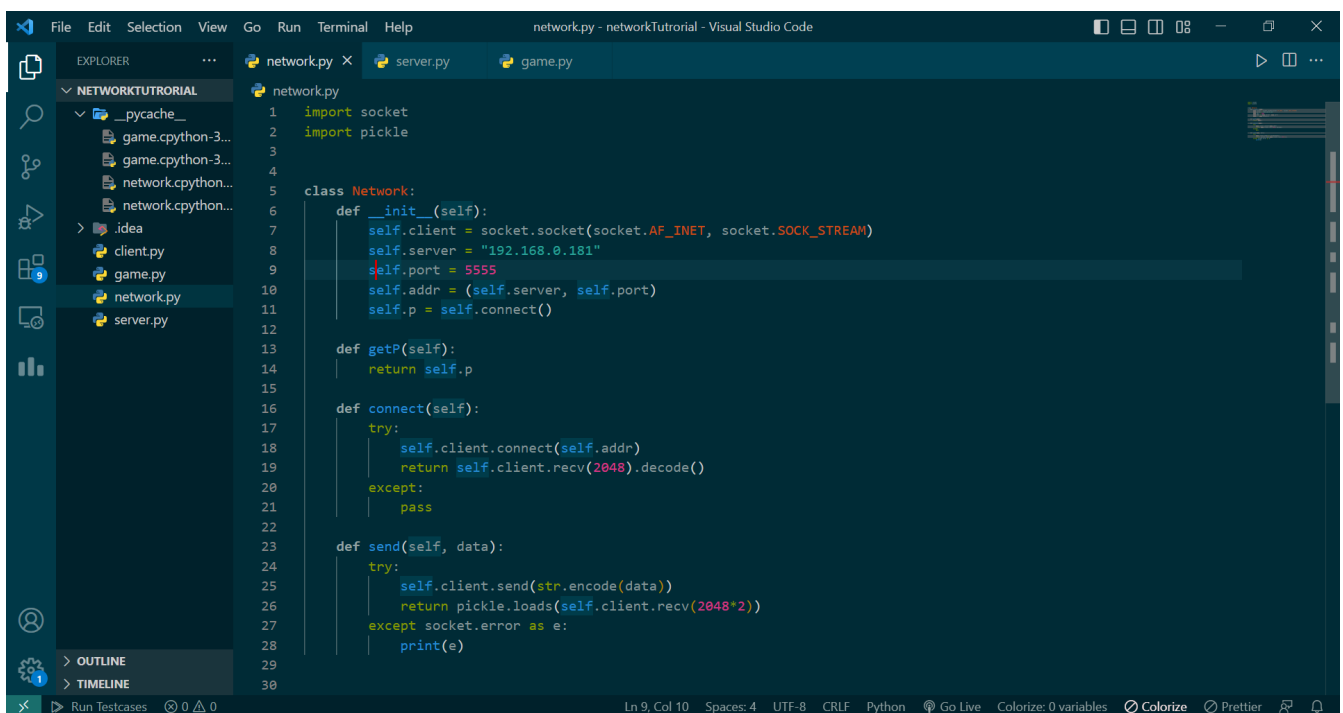
This diagram shows the overall flow of the game application from start to finish. The user starts the game, which displays a menu of options. The user selects a game option, which then connects to the server. The user then waits for an opponent to connect and for the game to begin. Once the game starts, the user plays a round of rock paper scissors with their opponent. The application then displays the result of the round, and waits for the next round to begin. Once the game is over, the application displays the final results and the user exits the game.

## 4. CODE AND SNAPSHOT



```
1 class Game:
2     def __init__(self, id):
3         self.p1Went = False
4         self.p2Went = False
5         self.ready = False
6         self.id = id
7         self.moves = [None, None]
8         self.wins = [0,0]
9         self.ties = 0
10
11     def get_player_move(self, p):
12         """
13         :param p: [0,1]
14         :return: Move
15         """
16         return self.moves[p]
17
18     def play(self, player, move):
19         self.moves[player] = move
20         if player == 0:
21             self.p1Went = True
22         else:
23             self.p2Went = True
24
25     def connected(self):
26         return self.ready
27
28     def bothWent(self):
29         return self.p1Went and self.p2Went
```

fig 3: Code snippet



```
1 import socket
2 import pickle
3
4
5 class Network:
6     def __init__(self):
7         self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         self.server = "192.168.0.181"
9         self.port = 5555
10         self.addr = (self.server, self.port)
11         self.p = self.connect()
12
13     def getP(self):
14         return self.p
15
16     def connect(self):
17         try:
18             self.client.connect(self.addr)
19             return self.client.recv(2048).decode()
20         except:
21             pass
22
23     def send(self, data):
24         try:
25             self.client.send(str.encode(data))
26             return pickle.loads(self.client.recv(2048*2))
27         except socket.error as e:
28             print(e)
```

fig 4: Code snippet

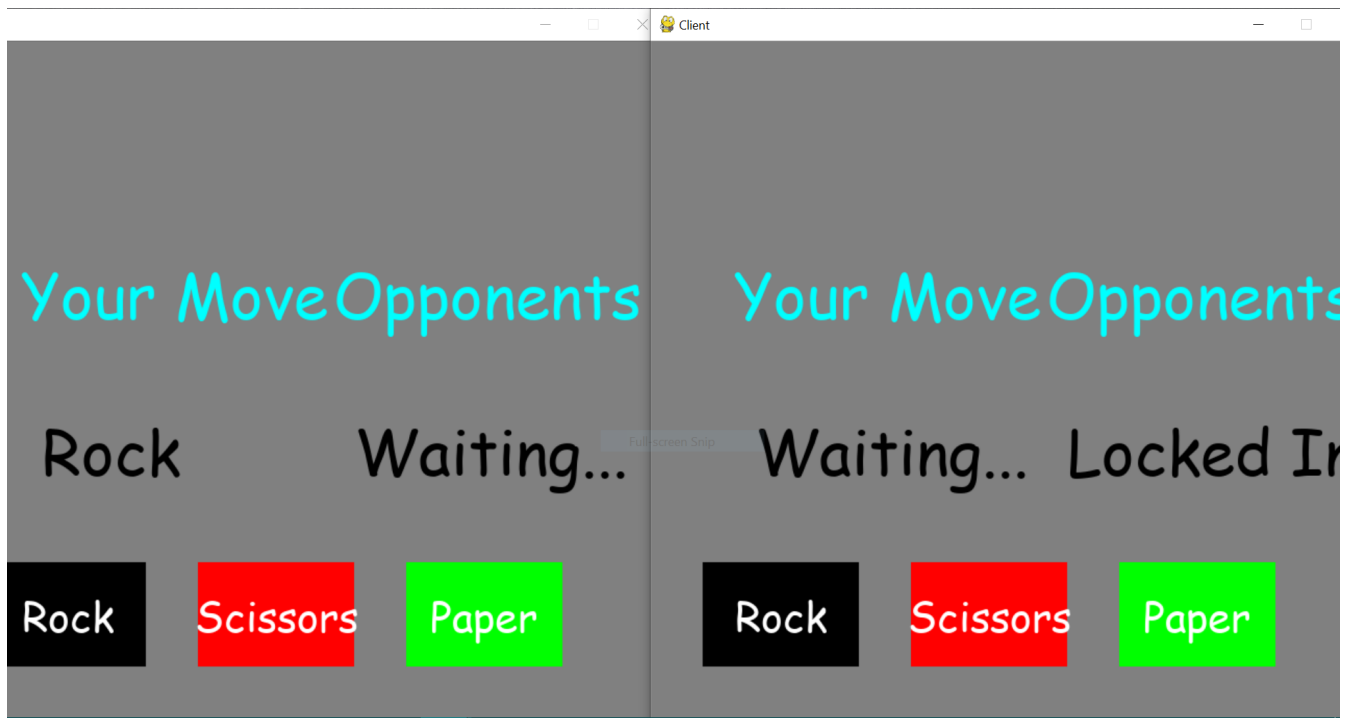


fig 5 : Output

## **5. RESULT**

The Rock Paper Scissors is a multiplayer game that demonstrates the use of network programming and graphics programming. The game provides an interactive and engaging experience for players, allowing them to compete against each other over a network connection. The use of sockets and networking programming makes it possible for players to communicate their choices to the server and receive their opponent's choices seamlessly.

The graphical interface developed using Pygame provides a visually appealing display of the game options and the score of each player. The game's logic is implemented in Python, which determines the winner of each round based on the standard Rock Paper Scissors rules.

The project showcases the use of Python, Pygame, and networking programming to create an interactive game that can be played by multiple players. It provides a foundation for future development and enhancements to include additional features, such as artificial intelligence, sound effects, and customization options. Overall, the project is a successful implementation of a Rock Paper Scissors game with networking capabilities and provides an excellent learning opportunity for students and aspiring game developers.

## **6. CONCLUSION AND FUTURE SCOPE**

In conclusion, the Rock Paper Scissors game developed using Python and Pygame provides an excellent example of how networking can be used to create interactive games.

Moreover, the game's graphical interface developed using Pygame provides an immersive and engaging experience for players, making the game enjoyable to play. The game's use of the standard Rock Paper Scissors rules ensures that the game is easy to learn and understand, making it accessible to a wide range of players.

In terms of future scope, the game can be further improved by incorporating additional features such as a leaderboard, sound effects, and customizable avatars for players. The game could also be expanded to include other classic games such as Tic-Tac-Toe or Chess, which could increase its appeal and attract a broader audience.

Overall, the Rock Paper Scissors game developed using Python and Pygame provides an excellent foundation for future development and is a great example of how networking can be used to create engaging games.