TCP CLIENT-SERVER PYTHON QUIZ

A COURSE PROJECT REPORT

By

PRAKHAR PARAKH (RA2011033010010)
SANA ANGEL ROSE (RA2011033010022)
JYOTI YADAV (RA2011033010040)
SANSKRITI SINHA (RA2011033010041)

Under the guidance of

Dr. Hariharan B

In partial fulfillment for the Course of

18CSC302J - COMPUTER NETWORKS in CINTEL

FACULTY OF ENGINEERING AND

TECHNOLOGY SRM INSTITUTE OF SCIENCE

AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2022

TABLE OF CONTENTS

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**TCP Client-Server Python Quiz** " is the bonafide work of **PRAKHAR PARAKH (RA2011033010010), SANA ANGEL ROSE (RA2011033010022), JYOTI YADAV (RA2011033010040), SANSKRITI SINHA (RA2011033010041)** who carried out the project work under my supervision.

SIGNATURE

Mrs. Annie Uthra
HOD

CINTEL
SRM Institute of Science and Technology

SIGNATURE

Dr. Hariharan B
Assistant Professor
Computational Intelligence
CINTEL

ABSTRACT

A network has been designed for conducting quiz competitions which can have multiple users or participants. The organization hosts a quiz competition on a server which is accessible to internet users using the public IP address.

A network was designed using the Python programming language. The basic requirements were tested. A server was set up, which can have connections with multiple users. Clients or users can play a quiz online and each client can view the score of the other participants. Also, a buzzer is provided there whoever will press it first will answer first.

If the user guesses the correct answer he gets +1 score and if we guess wrong one then he gets -1 score. In the end, the winner is broadcast with the score.

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 are highly thankful to our Course project **Faculty Dr.Hariharan B**, Assistant Professor CINTEL, for his assistance, timely suggestions, for his constant encouragement and support and guidance throughout the duration of this course project.

We extend my gratitude to our **HOD Mrs. Annie Uthra** (CINTEL) 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, we thank the almighty for showering his blessings on me to complete my Course project.

INTRODUCTION

Scenario Description

A network has been designed for conducting quiz competitions which can have multiple users or participants. The organization hosts a quiz competition on a server which is accessible to internet users using the public IP address.

A network was designed using the Python programming language. The basic requirements were tested. A server was set up, which can have connections with multiple users. Clients or users can play a quiz online and each client can view the score of the other participants. Also, a buzzer is provided there whoever will press it first will answer first.

If the user guesses the correct answer he gets +1 score and if we guess wrong one then he gets -1 score. In the end, the winner is broadcast with the score.

LITERATURE SURVEY

The protocols at the transport layer of the OSI reference model provide an end-to-end data delivery service for application processes to exchange messages over the Internet. Protocols operating in the Transport layer use the services of the Internet Protocol (IP) to deliver messages. Many protocols exist at this layer and one of the most important protocols at this layer is Transmission Control Protocol (TCP) for the delivery of data between the Application Layer and the Internet Layer. The shortcoming of IP to guarantee the delivery of datagrams is overcome by TCP by setting up a virtual circuit between the communicating parties.

The design of TCP was specifically to provide reliable end-to-end delivery of data over an unreliable medium. The unreliable medium of transport is the Internet Protocol (IP). TCP is a connection-oriented protocol with guaranteed delivery of transport layer segments and error detection mechanism.

Although IP does the major work in moving datagrams around the network as needed, TCP is important to make sure the data inside of the IP datagram is correct. Critical applications such as e-mail and Telnet require reliable and guaranteed delivery of data which is made possible by a virtual circuit that TCP sets up whenever two applications need to communicate. Before communications begin using TCP, the two applications have to agree on some basic parameters before segments can be sent. This is done in a manner referred to as a three-way handshake.

In the three-way handshake before the sender can start sending, rules for data transmission are negotiated. The first part of a three-way handshake begins with the sender transmitting a TCP segment with the Synchronization ("SYN") bit set. The TCP sequence number is the value being negotiated here. The recipient responds with a TCP segment with both the synchronization and acknowledgement bits set as "SYN/ACK". This is the second part of the three-way handshake. The sender then responds with an ACK, and the three-way handshake is complete. In tearing down the communication channel, TCP uses the FIN ("finish") bit to bring the channel down when the communication is closed.

TCP performs error detection and error recovery by using a sequence number and an acknowledgement number ("ACK") in the TCP header. The sender waits for a positive message from the recipient that the data was received and if that message isn't received, the data is retransmitted. This process is known as Positive Acknowledgement with Retransmission (PAR).
TCP performs error detection and error recovery by using a sequence number and an acknowledgement number ("ACK") in the TCP header. The sender waits for a positive message from the recipient that the data was received and if that message isn't received, the data is retransmitted. This process is known as Positive Acknowledgement with Retransmission (PAR).

A client-server application is a category of a distributed system made up of both client and server software. The client-server application provides an enhanced way to share the workload. The client process continuously launches a connection to the server, while the server process still expects for requests from any client. A client is a computer hardware device with software that accesses a service made available by a server. A server is a computer, a dedicated software run on it and provides services to serve the needs of other machines. Depending on the service that is running, it could be a file server, database server, home media server, print server, web server.

Socket Programming

Socket programming 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. They are the real backbones behind web browsing. In simpler terms, there is a server and a client.

Client

A client application is a process or program that sends a job request to a server via the communication network. Those jobs request the server to perform a particular task, such as looking up a record in a database or returning a portion or customized report. Examples of clients are a web browser, thin client, remote desktop, emulator, front-end application, mobile app, etc.

Functions used :

1. Using socket(), create the socket descriptor
2. Using connect(), connect to the remote server
3. Using read(),write(), communicate with the server
4. Using close(), end communication by closing socket descriptor

Server

The server is a collection of the programme, listens for client requests that are transmitted via the communication network. Servers perform actions such as database queries or reading files. Server processes typically run on dominant PCs, workstations or mainframe computers.

Functions used :

Functions used :

1. Using socket(), create the socket descriptor.
2. Using connect(), connect to the remote server.
3. Using read(),write(), and communicating with the server.
4. Using close(), end communication by closing socket descriptor
5. Using bind(), bind the socket to the server address.
6. Using listen(), put the server socket in a passive mode, where it waits for the client to approach the server to make a connection.
7. Using accept()- accept incoming connection.

REQUIREMENTS

Requirement Analysis

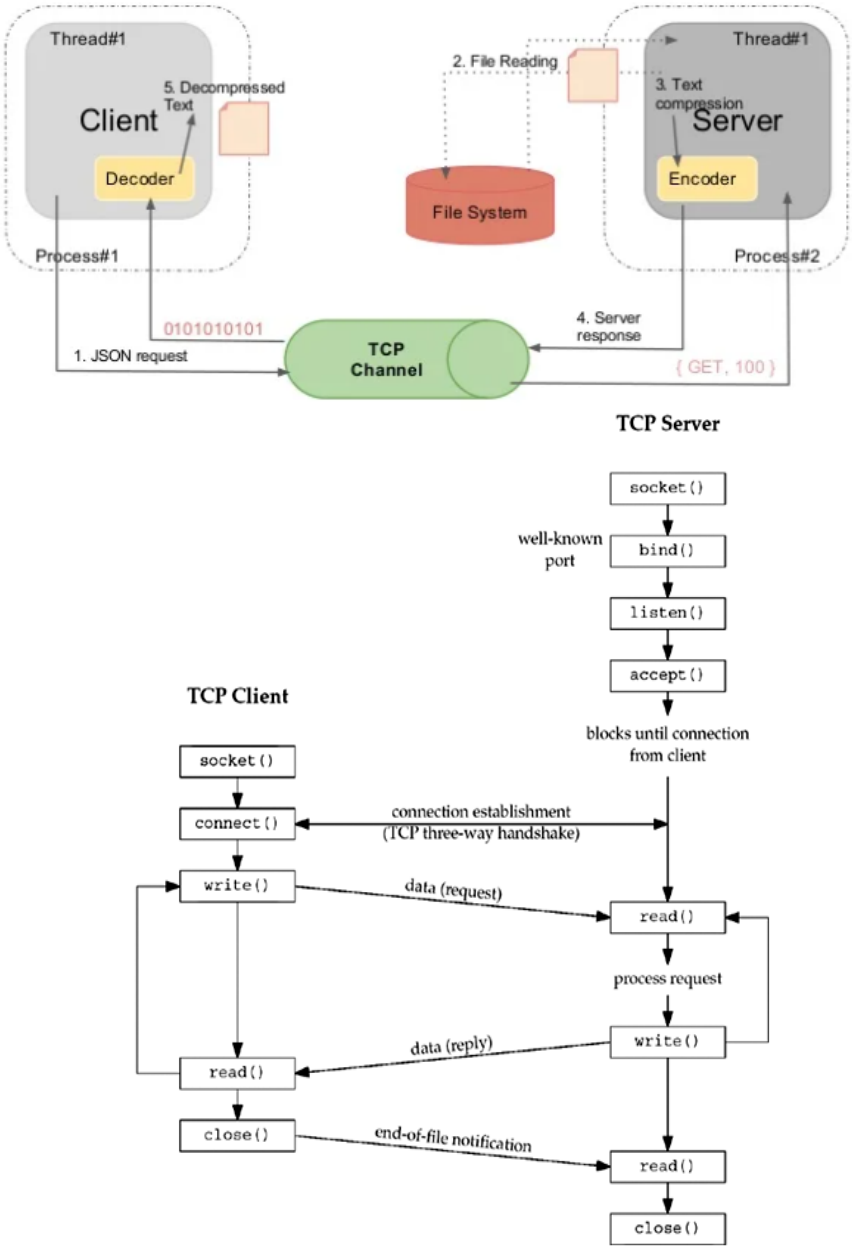
From the given scenario, we draw the following requirements:

1. Identifying the appropriate platform which would be used (Python, Terminal)
2. Users on the internet should be able to access only https on the server.
3. Users on the internet should have access only to the public IP address of the server and not the private IP address.
4. The users in the organization should have full access to the server.
5. TCP/IP Network design with Socket programming using Python.

ARCHITECTURE AND DESIGN

Network Architecture

The network architecture is as follows:



The architecture consists of three major networks:

- Company Network(s)
- Public Internet or multiple clients (here it is three)
- Network maintained by the Internet Service Provider

These networks are interconnected with each other with varying degrees.

IMPLEMENTATION

1. Code Implementation

Server :

```
import socket
import select
from _thread import *
import sys
import time
import random


server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

if len(sys.argv) != 3:
   print("Print in the following order : script, IP address, port number")
   exit()

IP_address = str(sys.argv[1])
Port = int(sys.argv[2])

server.bind((IP_address, Port))
server.listen(3)

list_of_clients = []

Q = [" What is the Italian word for PIE? \n a.Mozarella b.Pasty c.Patty d.Pizza",
    " Water boils at 212 Units at which scale? \n a.Fahrenheit b.Celsius c.Rankine d.Kelvin",
    " Which sea creature has three hearts? \n a.Dolphin b.Octopus c.Walrus d.Seal",
    ]

A = ['d', 'a', 'b']

Count = []
client = ["address", -1]
bzr = [0, 0, 0]


def clientthread(conn, addr):
    conn.send(bytes("Hello Genius!!!\n Welcome to this quiz! Answer any 5 questions correctly before your opponents do\n Press any key on the keyboard as a buzzer for the given question\n",'utf-8'))
   # Intro MSG
   while True:
      message = conn.recv(2048).decode()
      if message:
         if bzr[0] == 0:
            client[0] = conn
```

```
            bzr[0] = 1
            i = 0
            while i < len(list_of_clients):
                if list_of_clients[i] == client[0]:
                    break
                i += 1
            client[1] = i

        elif bzr[0] == 1 and conn == client[0]:
            bol = message[0] == A[bzr[2]][0]
            print(A[bzr[2]][0])
            if bol:
                broadcast("player" + str(client[1] + 1) + " +1" + "\n\n")
                Count[i] += 1
                if Count[i] == 3:
                    broadcast("player" + str(client[1] + 1) + " WON" + "\n")
                    end_quiz()
                    sys.exit()

            else:
                broadcast("player" + str(client[1] + 1) + " -1" + "\n\n")
                Count[i] -= 1
            bzr[0] = 0
            if len(Q) != 0:
                Q.pop(bzr[2])
                A.pop(bzr[2])
            if len(Q) == 0:
                end_quiz()
            quiz()

        else:
            conn.send(bytes(" player " + str(client[1] + 1) + " pressed buzzer first\n\n",'utf-8'))
    else:
        remove(conn)


def broadcast(message):
    for clients in list_of_clients:
        try:
            clients.send(bytes(message,'utf-8'))
        except:
            clients.close()
            remove(clients)


def end_quiz():
    broadcast("Game Over\n")
    bzr[1] = 1
    i = Count.index(max(Count))
    broadcast("player " + str(i + 1) + " wins!! by scoring " + str(Count[i]) + " points.")
    for x in range(len(list_of_clients)):
        list_of_clients[x].send(bytes("You scored " + str(Count[x]) + " points.",'utf-8'))
```

```
    # server.close()


def quiz():

    if len(Q) != 0:
        bzr[2] = random.randint(1, 10000) % len(Q)
        for connection in list_of_clients:
            connection.send(bytes(Q[bzr[2]],'utf-8'))


def remove(connection):
    if connection in list_of_clients:
        list_of_clients.remove(connection)


while True:
    conn, addr = server.accept()
    list_of_clients.append(conn)
    Count.append(0)
    print(addr[0] + " connected")

    start_new_thread(clientthread, (conn, addr))
    if (len(list_of_clients) == 3):
        quiz()
conn.close()
server.close()
```

Client :

```
import socket
import select
import sys

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
if len(sys.argv) != 3:
    print ("Print in the following order : script, IP address, port number")
    exit()

IP_address = str(sys.argv[1])
Port = int(sys.argv[2])
server.connect((IP_address, Port))

while True:

    sockets_list = [sys.stdin, server]

    read_sockets,write_socket, error_socket = select.select(sockets_list,[],[])

    for socks in read_sockets:
        if socks == server:
```

```
            message = socks.recv(2048).decode()
            print (message)
        else:
            message = sys.stdin.readline()
            server.send(bytes(message,'utf-8'))
            sys.stdout.flush()
server.close()
sys.exit()
```

2. Address Table

The address table is as follows:

Device	**Interface**	**Address**
Server	Port Number	127.0.0.1
Client 1	Port Number	127.0.0.1
Client 2	Port Number	127.0.0.1
Client 3	Port Number	127.0.0.1
Port Number	9999	-

The Access Control List contains the entire broadband network. Any request from that Server is translated to the private IP of the server.

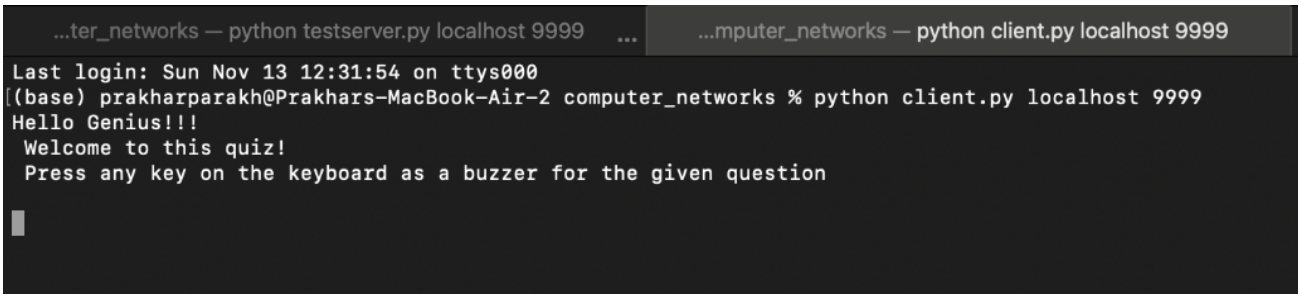
Socket Programming is used on the Server to connect to the Clients.

RESULTS AND DISCUSSION

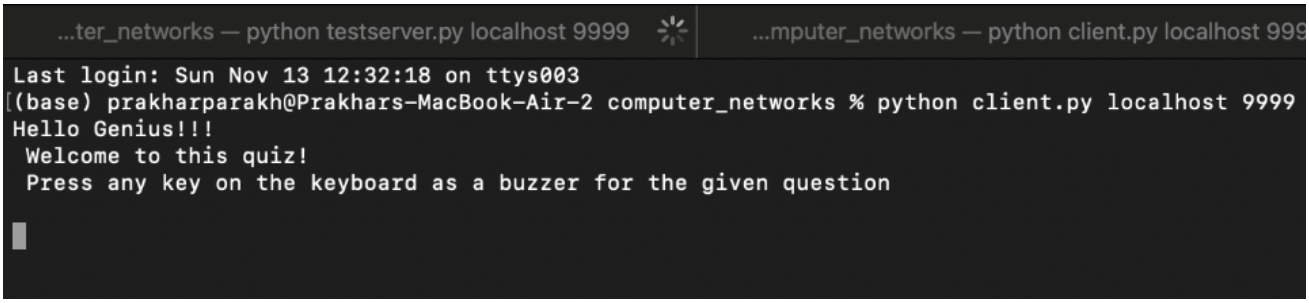
1.1 Connection Check

The network connections were checked:

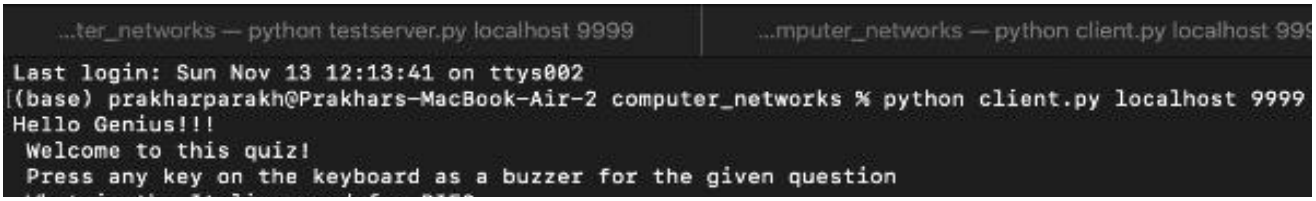
Client 1 connected to the server…



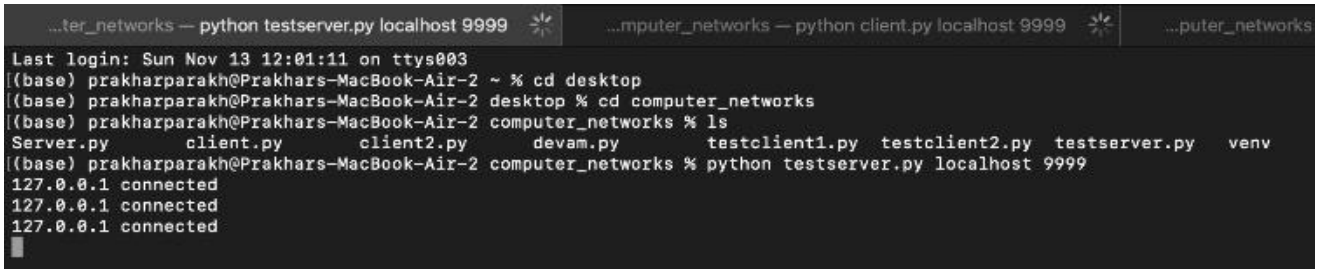
Client 2 connected to the server…



Client 3 connected to the server…

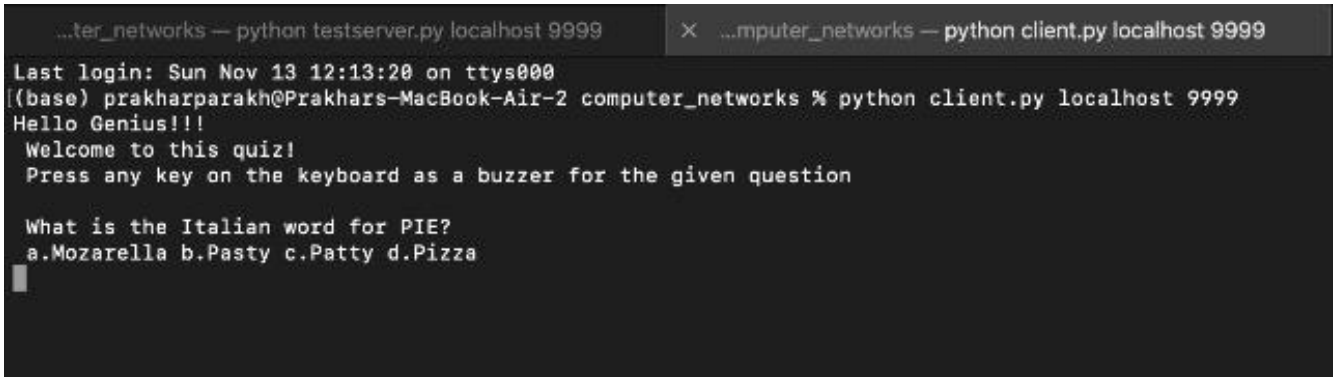


Now, all 3 Clients are connected to the server



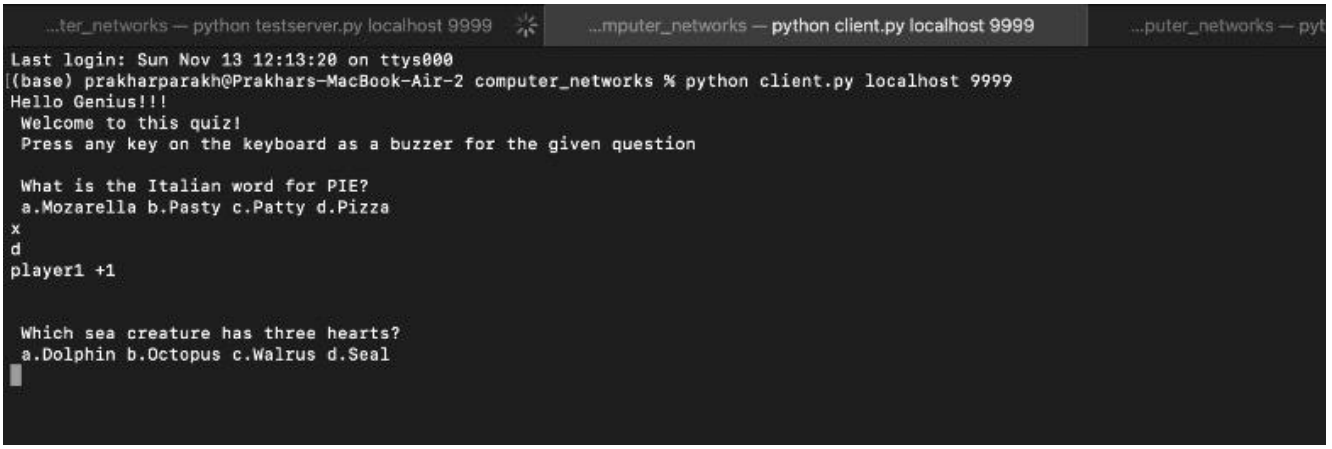
6.2 Conducting Quiz

The server asks the first question:

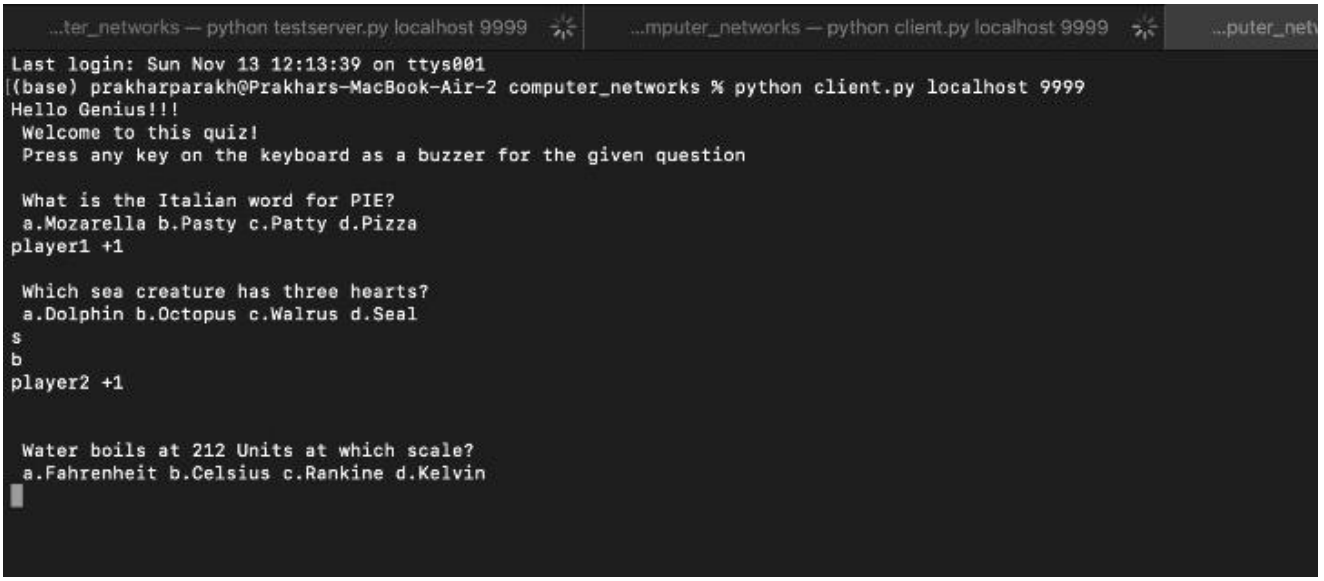


Client 1 pressed the buzzer first and gave the correct answer and gained +1 point.

Then, immediately after that second question is being asked

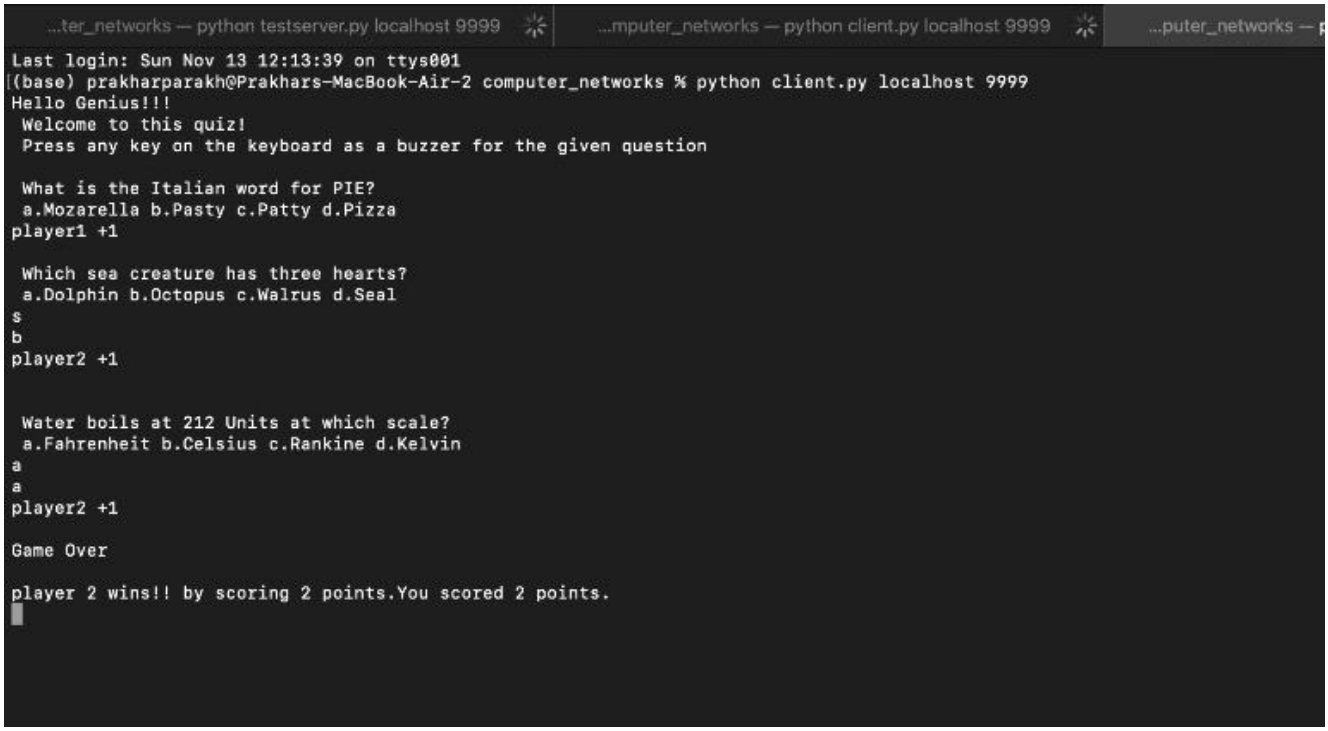


Now, Client 2 pressed the buzzer first and gave the correct answer and gained +1 point followed up by the third question.



Again, Client 2 pressed the buzzer first and gave the correct answer and gained +1 point.

The Game Ends, Player 2 wins the game.



CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, we created a game quiz between the server and multiple clients. The server is hosted by an organization. Clients have respected IP addresses and multiple clients can play at one time. If client 1 presses the buzzer first, he gets a chance to answer first and if client answers correctly they get +1 or else -1 is deducted from the result. Same procedure followed for Client 2 and 3. At the end of the quiz session, whichever client will win, it will be displayed on each client's interface.

For future enhancement, we can displayresult graph in the end.
Also, we can store the data i.e. the questions of the user in MySQL, moreover we can fetch the data from MySQL to the client's interface.

REFERENCES

1. https://medium.com/
2. https://geeksforgeeks.com/
3. Microsoft Bing Images
4. YT Socket Programming