**Real-Time Multi-Player Quiz Game**

**implemented using Python.**

Mini Project Report of

Computer Networks Lab (CSE 3162)

**Report**

Student 1: Devanshu Singh(Reg. no. 210905314)

Student 2: Vishal Puneyani(Reg. no. 210905037)

Student 3: Gauri Saxena (Reg. no. 210905346)

Student 4: Kuber Lamba(Reg. no. 210905070)

**Abstract**

This project introduces an intricately designed multiplayer quiz game using Python, featuring a diverse array of functionalities for an immersive user experience. The game incorporates multiple question genres, an authentication system, question-specific timers, a results page, and in-depth tracking of correct and incorrect answers.

Built on Python, the game leverages well-established libraries like Flask for web development and Socket.IO for seamless real-time communication among players. The authentication page ensures a secure entry point, enabling players to either create accounts or securely log in.

A standout element of this quiz game is its expansive question database, covering various genres such as general knowledge, science, history, and more. This diversity ensures an inclusive and engaging experience, catering to players with varied interests.

The inclusion of a timer for each question introduces an element of urgency, prompting players to think on their feet and make decisions swiftly. The timer seamlessly integrates into the game interface, intensifying the competitive aspect of the multiplayer experience.

Upon completing the quiz, players are redirected to a results page offering a comprehensive overview of their performance. This page includes the total count of correct and incorrect answers, a percentage score, and a detailed breakdown of scores by category. This granular feedback allows players to assess their strengths and weaknesses across different subjects.

Moreover, the game incorporates a leaderboard, showcasing the scores of all players actively participating in the multiplayer session. This feature fosters healthy competition and motivates players to aim for the top spot on the leaderboard.

To summarize, the Comprehensive Multi-Genre Multiplayer Quiz Game in Python provides a captivating and educational experience. By integrating a variety of questions, secure authentication, timed challenges, and detailed performance insights, the game aims to engage users with different interests and skill levels in a dynamic multiplayer gaming environment.

i

**Contents**

[**1 Introduction** **1**](#_Toc21118)

[**2 Literature Review** **1**](#_Toc21119)

[2.1 Socket Programming: An Overview 1](#_Toc21120)

[2.2 Tkinter for Graphical User Interface (GUI) 1](#_Toc21121)

[2.3 Client-Server Architecture 1](#_Toc21122)

[2.4 Concurrency and Multithreading 1](#_Toc21123)

[2.5 Data Management with Pandas 2](#_Toc21124)

[2.6 Conclusion 2](#_Toc21125)

[**3 Methodology** **2**](#_Toc21126)

[3.1](#_Toc21127) [Project Planning and Requirements Analysis](#_Toc21130)2

[3.2](#_Toc21128) [Technology Selection](#_Toc21131)2

3.3 [Database Design](#_Toc21132)........................................................................................2

3.4 Authentication System...............................................................................2

[**4 Implementation Details** **2**](#_Toc21129)

[4.1 Server 2](#_Toc21130)

[4.2](#_Toc21131) [Client](#_Toc21134)3

[**5 Code Implementation** **4**](#_Toc21133)

[5.1 Client Code 4](#_Toc21134)

[5.2 Server Code 6](#_Toc21135)

[**6 Results and Analysis** **9**](#_Toc21136)

[**7 Discussion** **10**](#_Toc21137)

[**8 Conclusion** **10**](#_Toc21138)

[**9 Future Work** **11**](#_Toc21139)

[**10 References** **12**](#_Toc21140)

ii

# Introduction

This project introduces a dynamic Multiplayer Quiz Game developed in Python, incorporating the use of the TCP protocol for robust communication between players. The game features an extensive question bank encompassing multiple genres, ensuring a diverse and engaging experience for participants. A secure authentication page ensures a protected entry point, while a timer for each question adds an element of urgency, stimulating quick thinking and decision-making.

The implementation of the TCP protocol enhances real-time communication, creating a seamless and responsive multiplayer environment. The game provides participants with a detailed results page at the end of each session, offering insights into their performance, including the count of correct and incorrect answers. This feature facilitates a personalized learning experience, allowing players to identify their strengths and areas for improvement across different subjects. In summary, the Multiplayer Quiz Game not only leverages Python's versatility but also utilizes the TCP protocol to create an interactive and educational gaming platform, fostering a collaborative and competitive atmosphere among participants.

# Literature Review

## **Socket Programming: An Overview**

Socket programming is a fundamental concept in computer networking, allowing communication between different processes on the same or different machines. In the context of the Multiclient Quiz App, sockets are used to establish a reliable connection between the server and clients. This facilitates the exchange of quiz questions, options, and answers in real-time.

## **Tkinter for Graphical User Interface (GUI)**

Tkinter is a standard GUI toolkit in Python, widely used for creating interactive and visually appealing user interfaces.

## **Client-Server Architecture**

Client-server architecture is a well-established model in networking where clients request services or resources from a central server.

## **Concurrency and Multithreading**

Concurrency is a key aspect of software development, allowing multiple tasks to execute seemingly simultaneously.

## **Data Management with Pandas**

Pandas is a data manipulation and analysis library in Python, commonly used for handling structured data, such as Excel files. Role of Pandas in efficiently reading and managing quiz data from external sources. It enables the seamless integration of data into the application.

## Conclusion

In conclusion, the development of this Multiplayer Quiz Game, grounded in Python and fortified by the TCP protocol, represents a successful fusion of technology and education. The incorporation of real-time communication through TCP ensures a seamless multiplayer experience, while the diverse genre of questions, secure authentication, and timed challenges contribute to an engaging and intellectually stimulating environment. The project's focus on providing detailed performance feedback fosters a commitment to continuous learning, allowing participants to not only compete but also enhance their knowledge across various subjects. This game serves not only as a testament to the versatility of Python in game development but also as a bridge connecting entertainment and education, exemplifying the potential of interactive technology to enrich and engage users in a collaborative learning journey.

# Methodology

1. Project Planning and Requirements Analysis:
   * Conduct an in-depth analysis of project requirements, including the desired features and functionalities.
   * Define the scope, objectives, and constraints of the Multiplayer Quiz Game.
2. Technology Selection:
   * Choose Python as the primary programming language for its versatility and extensive libraries.
   * Select the TCP protocol for real-time communication, ensuring seamless multiplayer interactions.
3. Database Design:
   * Design a comprehensive question database covering various genres to provide a diverse and engaging quiz experience.
   * Implement a secure and efficient database system to store user authentication details, questions, and performance metrics.
4. Authentication System:
   * Develop a secure authentication page to manage user accounts, ensuring a protected and personalized gaming environment.
5. Game Logic Implementation:
   * Create the core game logic responsible for handling multiplayer interactions, question delivery, and timer management.
   * Implement algorithms for scoring, tracking correct and incorrect answers, and generating real-time performance metrics.
6. TCP Protocol Integration:
   * Incorporate the TCP protocol to facilitate real-time communication between players, ensuring a responsive and synchronized multiplayer experience.
   * Implement error-handling mechanisms to address potential issues related to network communication.
7. User Interface (UI) Design:
   * Design an intuitive and visually appealing user interface for the quiz game, including question displays, timers, and results pages here we have incorporated Tkinter.
   * Ensure a seamless user experience with clear navigation and responsive design elements.
8. Documentation and User Guides:
   * Prepare comprehensive documentation detailing the system architecture, code structure, and deployment procedures.
   * Create user guides to assist players in navigating the game, understanding rules, and interpreting results.

By following this methodology, the project aims to deliver a robust, feature-rich Multiplayer Quiz Game that combines entertainment with educational value.

# Implementation Details

The implementation involves two main components: the server and the client.

#### **Server Implementation**

* The server initializes a socket and binds it to a specific address and port.
* Listens for incoming connections from clients.
* Accepts client connections and spawns a new thread for each client to handle concurrent quizzes.
* Reads quiz questions from an Excel file using Pandas.
* Communicates with clients, sending questions and receiving answers.
* Calculates and updates scores based on correct answers.

#### **Client Implementation**

* The client initializes a socket and connects to the server.
* Participates in the quiz by receiving questions and sending answers.
* Displays the quiz interface using Tkinter.
* Handles different types of questions (multiple choice, text entry).
* Submits answers to the server for evaluation.

# Code Implementation

**SERVER:**

import socket

import threading

import time

import pandas

import sys

HOST="10.86.3.121"

PORT=8080

BUFFERSIZE=1024

CURR\_CLIENT\_NO = 0

TOT\_CLIENT\_NO=0

SERVER=None

playerName=""

results=[]

df=pandas.read\_excel("Questions.xlsx")

score=0

def select\_questions(sheet\_name,client):

global df,questions,score

df=pandas.read\_excel("Questions.xlsx",sheet\_name=sheet\_name)

subset=df.sample(n=3)

questions=subset["Question"].values

option\_1=subset["Option\_1"].values

option\_2=subset["Option\_2"].values

option\_3=subset["Option\_3"].values

option\_4=subset["Option\_4"].values

answer=subset["Answer"].values

start\_time=time.time()

for i in range(0,len(questions),1):

combined=f"{questions[i]}|{option\_1[i]}|{option\_2[i]}|{option\_3[i]}|{option\_4[i]}|{answer[i]}"

print(f"Combined :{combined}")

client.send(bytes(combined,"utf8"))

user\_answer = client.recv(128).decode("utf8")

if user\_answer == answer[i]:

score+=1

print(f"Score : {score}")

end\_time=time.time()

total\_time=end\_time-start\_time

print(total\_time)

results.append({

"name":playerName,

"time":total\_time

})

def credential\_check(credentials):

global playerName

with open('Credentials.txt', 'r') as file:

lines = file.readlines()

for line in lines:

line = line.strip()

playerName,pwd=line.split('-')

if line==credentials:

return True

return False

def handle\_client(client):

client.send(bytes("Welcome to the quiz!","utf8"))

credentials=client.recv(1024).decode("utf8")

print(credentials)

if credential\_check(credentials):

client.send("1".encode("utf8"))

else:

client.send(bytes("0","utf8"))

sheet\_name=client.recv(1024).decode("utf8")

print(sheet\_name)

select\_questions(sheet\_name,client)

def Main():

global SERVER,results

global TOT\_CLIENT\_NO,CURR\_CLIENT\_NO

TOT\_CLIENT\_NO=int(input("Enter max number of players : "))

result=""

SERVER = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

SERVER.bind((HOST,PORT))

SERVER.listen(TOT\_CLIENT\_NO)

while CURR\_CLIENT\_NO<TOT\_CLIENT\_NO:

CURR\_CLIENT\_NO+=1

client,client\_address=SERVER.accept()

print("Connection "+str(CURR\_CLIENT\_NO)+" : "+client\_address[0])

client\_handler\_thr=threading.Thread(target=handle\_client,args=(client,))

client\_handler\_thr.start()

for i in results:

result=result+str(i["name"])+","+str(i["time"])+";"

print(result)

SERVER.close()

if \_\_name\_\_ == '\_\_main\_\_':

Main()

**CLENT:**

import socket

from threading import Thread

import time,sys,select

import tkinter

from tkinter import messagebox,ttk

import pandas

HOST="10.86.3.121"

PORT=8080

BUFFERSIZE=4096

remaining\_time = 60

question\_number = 1

df=pandas.read\_excel("Questions.xlsx")

# HOST=input("Enter Host address : ")

# PORT=int(input("Enter port number : "))

CLIENT = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)

CLIENT.connect((HOST,PORT))

def results():

print("we are here")

def next\_question(question\_number\_label,question\_label,button\_a,button\_b,button\_c,button\_d,answer\_entry,answer\_label):

global question\_number,remaining\_time

remaining\_time=60

if question\_number==3:

clear\_screen()

results()

question\_number += 1

question\_number\_label.config(text=f"Question {question\_number}")

answer=answer\_entry.get()

answer\_entry.delete(0,tkinter.END)

print(f"Your answered : {answer}")

CLIENT.send(bytes(answer,"utf8"))

question=CLIENT.recv(256).decode("utf8")

# print(f"Combined: {question}")

question,option\_1,option\_2,option\_3,option\_4,answer = question.split("|")

print(f"Question : {question}")

# print(f"{question} , {option\_1} , {option\_2} , {option\_3} , {option\_4} , {answer}")

question\_label.config(text=question)

if option\_2=="nan":

# answer\_label = tkinter.Label(frame, text="Type Answer", bg='#a2d2ff', fg="#000000", font=("Georgia", 16))

# answer\_entry = tkinter.Entry(frame, font=("Arial", 16))

answer\_label.grid(row=2, column=0)

answer\_entry.grid(row=2, column=1, pady=20)

else:

button\_a.config(text=option\_1)

button\_b.config(text=option\_2)

button\_c.config(text=option\_3)

button\_d.config(text=option\_4)

# answer\_label = tkinter.Label(frame, text="Enter answer", bg='#a2d2ff', fg="#000000", font=("Georgia", 16))

# answer\_entry = tkinter.Entry(frame, font=("Arial", 16))

answer\_label.grid(row=4, column=0)

answer\_entry.grid(row=4, column=1, pady=20)

next\_question\_button = tkinter.Button(frame, bg="#80ffdb", text="Next Question", font=('Helvetica', 16), borderwidth=1, relief="solid",padx=10,pady=10,command=lambda: next\_question(question\_number\_label,question\_label,button\_a,button\_b,button\_c,button\_d,answer\_entry,answer\_label))

next\_question\_button.grid(row=6, column=0, pady=10)

window.update\_idletasks()

def update\_timer(timer\_label):

global remaining\_time

if remaining\_time > 0:

remaining\_time -= 1

timer\_label.configure(text=f"Time Remaining: {remaining\_time} seconds")

window.after(1000, update\_timer,timer\_label)

else:

timer\_label.configure(text="Time's up!")

def questions\_screen():

global df

clear\_screen()

window.title('Quiz Application')

frame.configure(bg="#a2d2ff",height=600,width=800)

question\_number\_label = tkinter.Label(frame, bg="#80ffdb",text="Question 1", font=('Helvetica', 16), borderwidth=2, relief="sunken",padx=15,pady=15)

question\_number\_label.grid(sticky="w",row=0,column=0,padx=20,pady=20)

timer\_label = tkinter.Label(frame, bg="#80ffdb",text="Time Remaining: 60 seconds", font=('Helvetica', 16), borderwidth=2, relief="sunken",padx=15,pady=15)

timer\_label.grid(sticky="e",row=0,column=4,padx=20,pady=20)

update\_timer(timer\_label)

frame.grid\_columnconfigure(0, weight=2)

frame.grid\_columnconfigure(4, minsize=300, weight=0)

frame.pack\_propagate(False)

frame.pack(fill="both",expand=True)

question\_label = tkinter.Label(frame, bg="#a2d2ff", text="", font=('Helvetica', 20), padx=20,pady=20, wraplength=700)

question\_label.grid(row=1, column=0, columnspan=6, sticky="news",pady=10,padx=10)

print("Questions from server")

# for i in range(0,6,1):

question=CLIENT.recv(256).decode("utf8")

# print(f"Combined: {question}")

question,option\_1,option\_2,option\_3,option\_4,answer = question.split("|")

print(f"Question : {question}")

question\_label.config(text=question)

button\_a = tkinter.Button(frame, bg="#a2d2ff",text=option\_1, font=("Georgia", 12))

button\_b = tkinter.Button(frame, bg="#a2d2ff",text=option\_2, font=("Georgia", 12))

button\_c = tkinter.Button(frame, bg="#a2d2ff",text=option\_3, font=("Georgia", 12))

button\_d = tkinter.Button(frame, bg="#a2d2ff",text=option\_4, font=("Georgia", 12))

answer\_label = tkinter.Label(frame, text="Enter answer", bg='#a2d2ff', fg="#000000", font=("Georgia", 16))

answer\_entry = tkinter.Entry(frame, font=("Arial", 16))

if option\_1=="nan":

# answer\_label = tkinter.Label(frame, text="Type Answer", bg='#a2d2ff', fg="#000000", font=("Georgia", 16))

# answer\_entry = tkinter.Entry(frame, font=("Arial", 16))

answer\_label.grid(row=2, column=0)

answer\_entry.grid(row=2, column=1, pady=20)

else:

button\_a.grid(row=2, column=1,sticky="news", padx=10,pady=10)

button\_b.grid(row=2, column=4,sticky="news", padx=10,pady=10)

button\_c.grid(row=3, column=1,sticky="news", padx=10,pady=10)

button\_d.grid(row=3, column=4,sticky="news", padx=10,pady=10)

answer\_label.grid(row=4, column=0)

answer\_entry.grid(row=4, column=1, pady=20)

next\_question\_button = tkinter.Button(frame, bg="#80ffdb", text="Next Question", font=('Helvetica', 16), borderwidth=1, relief="solid",padx=10,pady=10,command=lambda: next\_question(question\_number\_label,question\_label,button\_a,button\_b,button\_c,button\_d,answer\_entry,answer\_label))

next\_question\_button.grid(row=6, column=0, pady=10)

window.update\_idletasks()

def on\_select(combo):

global df

selected\_item = combo.get()

print(f'Selected Quiz: {selected\_item}')

CLIENT.send(bytes(selected\_item,"utf8"))

questions\_screen()

def clear\_screen():

for widget in frame.winfo\_children():

widget.pack\_forget()

widget.destroy()

def select\_genre():

clear\_screen()

window.title('Genre Select')

header\_label = tkinter.Label(frame, text='Select your topic of choice', bg="#9d9d9d", font=("Georgia", 16))

header\_label.pack(pady=40)

combo = ttk.Combobox(frame, values=['Physics', 'Chemistry', 'Maths', 'English', 'Logical Reasoning', 'General Knowledge'])

combo.pack(pady=10)

combo.set('Chemistry')

select\_button = tkinter.Button(frame, text='Select', command=lambda: on\_select(combo))

select\_button.pack(pady=10)

combo['state'] = 'readonly'

def on\_enter(event):

login\_button.config(cursor="hand2")

def on\_leave(event):

login\_button.config(cursor="")

def login():

# username=username\_entry.get()

# password=password\_entry.get()

username="user1"

password="pass1"

credentials=f"{username}-{password}"

CLIENT.send(bytes(credentials,"utf8"))

data=CLIENT.recv(1024).decode("utf8")

if data=="1":

messagebox.showinfo(title="Login Success", message="You successfully logged in.")

select\_genre()

elif data=="0":

messagebox.showerror(title="Error", message="Invalid login.")

def start\_game():

data=CLIENT.recv(1024).decode("utf8")

print(data)

client\_thread=Thread(target=start\_game)

client\_thread.start()

window = tkinter.Tk()

window.title("Login Form")

window.geometry("800x600")

window.configure(bg='#9D9D9D')

frame = tkinter.Frame(bg='#9d9d9d')

login\_label = tkinter.Label(

frame, text="Login", bg='#9D9D9D', fg="#170F11", font=("Times New Roman", 30))

username\_label = tkinter.Label(

frame, text="Username", bg='#9D9D9D', fg="#FFFFFF", font=("Georgia", 16))

username\_entry = tkinter.Entry(frame, font=("Arial", 16))

password\_entry = tkinter.Entry(frame, show="\*", font=("Arial", 16))

password\_label = tkinter.Label(

frame, text="Password", bg='#9D9D9D', fg="#FFFFFF", font=("Georgia", 16))

login\_button = tkinter.Button(

frame, text="Login", bg="#170F11", fg="#FFFFFF", font=("Georgia", 16), command=login)

login\_label.grid(row=0, column=0, columnspan=2, sticky="news", pady=60)

username\_label.grid(row=1, column=0)

username\_entry.grid(row=1, column=1, pady=20)

password\_label.grid(row=2, column=0)

password\_entry.grid(row=2, column=1, pady=20)

login\_button.grid(row=3, column=0, columnspan=2, pady=30)

login\_button.bind("<Enter>", on\_enter)

login\_button.bind("<Leave>", on\_leave)

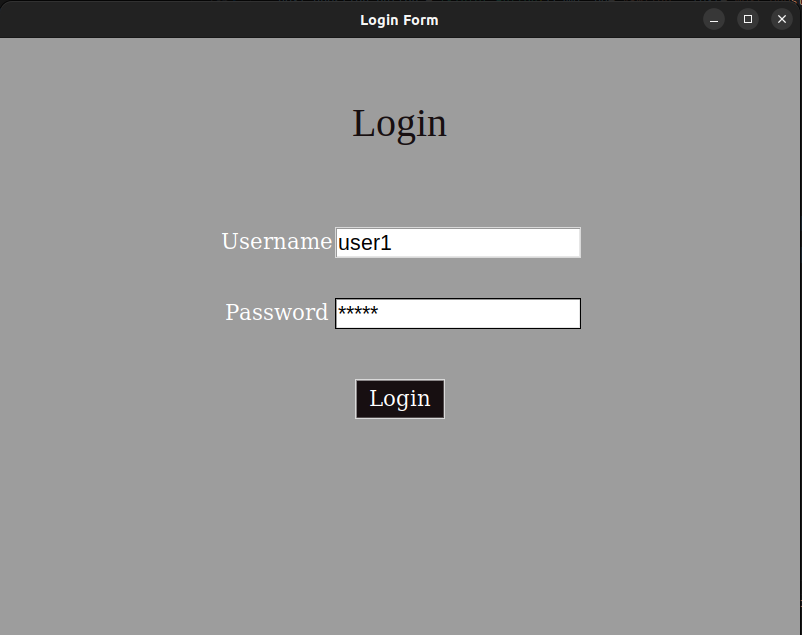
frame.pack()

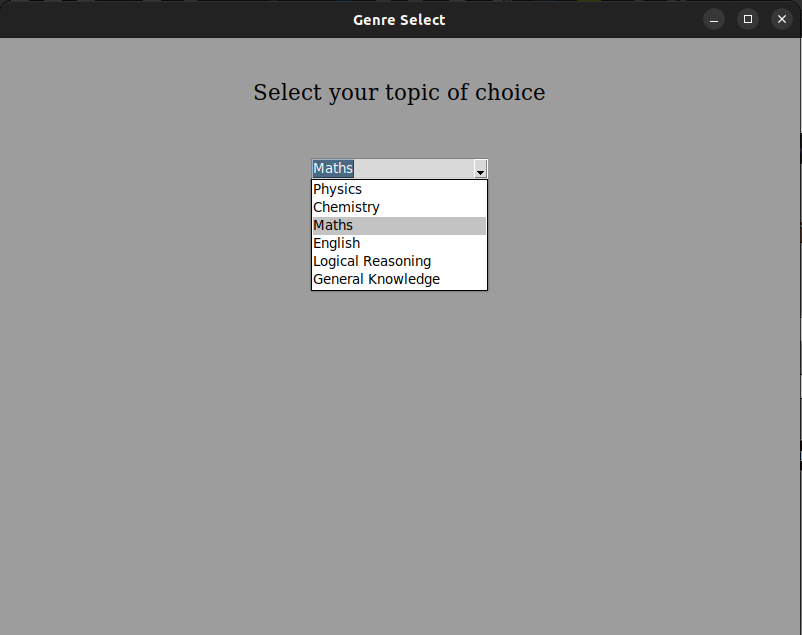
window.mainloop()

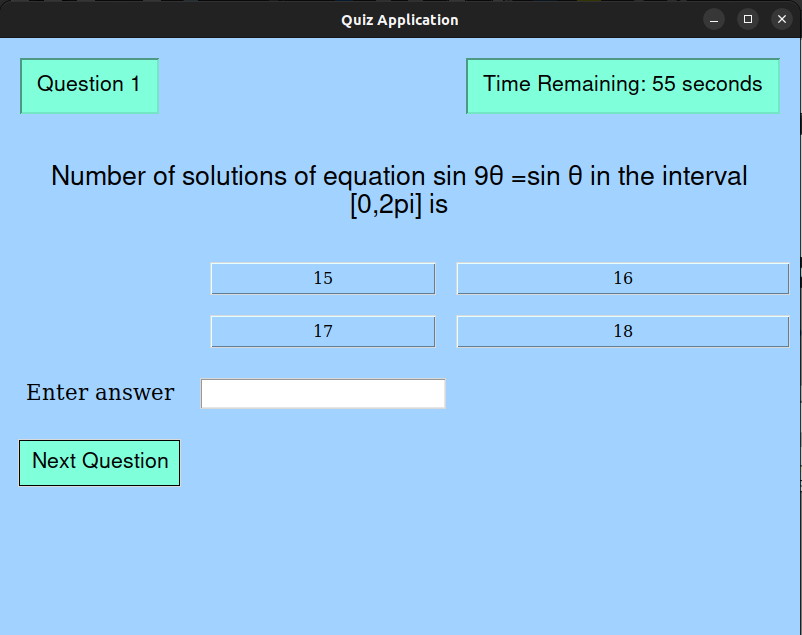
CLIENT.close()

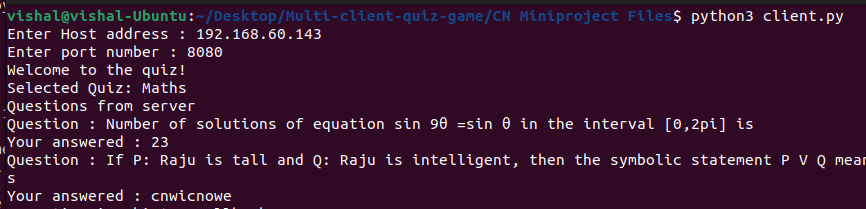
# Results and Analysis

The following points highlight the key outcomes of the project:









**Challenges Encountered:**

* Synchronization and Concurrency: Managing multiple simultaneous connections and ensuring synchronization between server and clients pose challenges in maintaining a seamless quiz experience.
* Security Concerns: Establishing poper connection between server and client for proper data flow

# Discussion

In summarizing the Multiplayer Quiz Game venture, the integration of Python and the TCP protocol has yielded a compelling and educational gaming venture. The project's success lies in achieving secure authentication, diverse question genres, and real-time multiplayer interaction. With timed challenges, detailed result pages, and performance tracking, the user experience is enriched, fostering both competition and continuous learning. This accomplishment underscores the potential of interactive technology to seamlessly blend entertainment and education, hinting at exciting prospects for collaborative and enlightening gaming encounters.

Looking forward, the Multiplayer Quiz Game serves as evidence of the dynamic landscape in interactive educational tools, emphasizing the importance of captivating platforms that ignite curiosity and facilitate learning. As technological advancements persist, this project establishes a foundation for ongoing exploration and refinement, signaling opportunities for future initiatives to harness gaming's potential as a robust educational instrument.

# Conclusion

In concluding the Multiplayer Quiz Game project, the successful integration of Python and the TCP protocol has resulted in an engaging and secure gaming experience. The project not only demonstrated the adaptability of network protocols but also underscored the importance of security measures within online multiplayer applications.

### Key Findings:

1. **TCP Protocol Effectiveness:**
   * The TCP protocol proved to be instrumental in establishing seamless real-time communication, enhancing the overall responsiveness and synchronization of the multiplayer gaming experience.
2. **Security Measures Significance:**
   * The implementation of a robust authentication system emphasized the critical role of network security, ensuring a protected and trustworthy user environment.
3. **Structured Communication Models:**
   * Timed challenges and performance tracking mechanisms highlighted the practical application of structured communication models in efficient data exchange, contributing to synchronized interactions.

### Contribution to Computer Networks Field:

The Multiplayer Quiz Game makes a noteworthy contribution to the field of Computer Networks by exemplifying:

1. **Adaptability of Networking Principles:**
   * The project showcases the adaptability of networking principles beyond conventional data exchange, providing a tangible example of their application in creating immersive and collaborative online experiences.
2. **Inspiration for Future Exploration:**
   * Serving as a practical demonstration, the project inspires future exploration and innovation at the convergence of computer networks and interactive gaming, offering a foundation for continued research in this dynamic intersection.

# Future Work

* Enhanced User Interface: Improve the graphical interface for a more user-friendly experience.
* Database Integration: Implement a database to store user scores and quiz statistics for future reference.
* Security Measures: Integrate security features to ensure the integrity and confidentiality of quiz data.
* Real-time Leaderboard: Implement a real-time leaderboard to display scores during the quiz.
* Mobile Compatibility: Extend the application to be compatible with mobile devices, increasing accessibility.

# References

1. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjkt-aekMqCAxVZe2wGHX6oBkEQFnoECBAQAQ&url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fsocket-programming-python%2F&usg=AOvVaw34RrdG2zyMfF1Lu2Ns4BxI&opi=89978449
2. https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python
3. Stack Overflow