

PES UNIVERSITY
EC Campus, Bengaluru
Department of Computer Science & Engineering



COMPUTER NETWORKS - UE21CS252B
4th Semester – E Section
ASSIGNMENT – 1

Online Quiz Application

Submitted to:

Dr. Geetha D
Associate Professor

Submitted By:

| | |
|-----------------------|--------------------|
| Name : Maryam Khan | SRN: PES2UG21CS283 |
| Name : Melvin A Gomez | SRN: PES2UG21CS293 |
| Name : Manasvi Varma | SRN: PES2UG21CS305 |

Table of Contents

| Sl. No. | Title | Page No |
|---------|-----------------------------------|---------|
| 1. | Abstract and Scope of the Project | 1 |
| 2. | Project Description | 2 |
| 3. | Software Requirements | 4 |
| 4. | Source Code | 6 |
| 5. | Sample Output | 10 |
| 6. | Conclusion | 13 |
| 7. | References | 14 |

1. Abstract and Scope of the Project

Abstract:

- The proposed project is an online quiz application that uses socket programming to allow multiple clients to connect to a server and take a quiz. The server will keep track of their scores and display the final results. The project aims to create a simple and user-friendly interface for conducting online quizzes.

Scope:

- The project includes the development of an online quiz application that allows multiple clients to connect to a server using socket programming.
- The application will include a server component and a client component. The server component will be responsible for managing the quiz questions, keeping track of the clients' scores, and displaying the final results. The client component will allow users to connect to the server, take the quiz, and view their scores.
- The application will support multiple-choice questions with a predefined set of answers. The server component will randomly select questions from a pool of questions and send them to the clients. The clients will answer the questions and send their responses to the server. The server will calculate the scores and display the final results to all the clients.
- The application will also include a feature to allow the server to broadcast messages to all connected clients, such as the start and end time of the quiz. The application will be developed using Python programming language and the socket programming module.
- The project aims to create a scalable and reliable application that can handle a large number of clients. The application will be tested on different operating systems and network configurations to ensure its compatibility and stability.

2.Project Description

The project includes two main classes, Server and Client, where Server waits for incoming client connections, and the Client communicates with the Server. The Server listens on a specific port and waits for incoming connections. Once the connection is established, the Server asks the Client for their username and starts sending multiple-choice quiz questions. The Client sends the answer to the Server, and the Server validates it and sends the next question until the quiz ends. After the quiz ends, the Server sends the final score to the Client.

This application uses TCP and socket programming which allows multiple users to connect to a server and communicate with each other by exchanging text messages. Unlike UDP, TCP is a Reliable protocol that guarantees complete delivery of messages and maintains a persistent connection between the client and server.

The Client class has three methods:

init(): initializes the class and tries to connect to the Server on the given IP address and port number. If the connection is unsuccessful, it terminates.

- serverName -> ip address default: 127.0.0.1
- serverPort -> port number default: 120000
- timer -> timer to get user input default : 10 seconds

GetUsername(): receives the message from the Server asking for the username and sends the username back to the Server.

MakeQuiz(): receives a question from the Server and waits for the user's answer. It then calculates the time elapsed between the question and the answer and sends the answer back to the Server. If the user takes more than a specified time, the answer is invalid, and the user gets no points for the question. It recursively calls itself until questions end.

The Server class also has two methods:

init(): initializes the class and creates a socket on the specified IP address and port number. It then waits for incoming connections from Clients.

MakeQuiz(): receives the username from the Client and sends multiple-choice quiz questions one by one. It then receives the Client's answers, validates them, and calculates the score. Once all the questions are answered, it sends the final score to the Client.

@param AF_INET : contains ipv4 address space

@param SOCK_STREAM : TCP connection based socket type

@param SOCK_DGRAM : UDP connection-less based socket type

- binds socket : associate socket with a specific network interface
- listen : enables server to socket.accept() listens socket (45)-numbers of unaccepted connection
- creates threads for all connected clients

Additionally, the project includes a dictionary of questions and answers that the Server uses to generate quiz questions. The project is designed to run on non-privileged ports (ports > 1023).

3. Software Requirements

These functions and APIs are used to create a basic **TCP client** that can send and receive messages from a **server**.

socket.socket: This is the Python API for creating a new socket object. The `AF_INET` argument specifies that the socket should use IPv4 addresses and the `SOCK_DGRAM` argument specifies that the socket should use the UDP protocol.

socket.bind: This function binds the socket to a specific address and port. In this case, the client is binding to the localhost address and a random port in the range 8000-9000.

socket.recvfrom: This function receives data from a socket and returns both the data and the address of the sender. In this case, the client is using it in a loop to continuously receive messages from the server.

socket.sendto: This function sends data to a specific address and port. In this case, the client is using it to send signup messages and chat messages to the server.

threading.Thread: This is a Python class for creating a new thread. The `target` argument specifies the function that the thread should run. In this case, the client is using it to create a separate thread for receiving messages from the server.

thread.start: This method starts the thread's execution. In this case, the client is using it to start the receive thread.

input: This is a built-in Python function for getting user input from the console.

exit: This is a built-in Python function for exiting the program.

random.randint: This function generates a random integer in a given range. In this case, the client is using it to choose a random port in the range 8000-9000.

encode: This method converts a string to bytes using a specified encoding. In this case, the client is using it to convert strings to bytes before sending them over the network.

decode: This method converts bytes to a string using a specified encoding. In this case, the client is using it to convert received messages from bytes to strings before printing them to the console.

setsockopt - This method is used to set the value of a socket option.

These methods are used to create a server object, set up the connection, and send and receive data between the server and the client. The threading module is used to create a new thread for each new client connection to handle multiple clients simultaneously

4.Source Code

Server.py

```
from socket import *
import threading

#locker = threading.Lock()
questions = {
    0: """
        Question 1 : Select the true choices according to following:
        1. TCP is Transmission Control Protocol.
        2. UDP is User Datagram Protocol.
        3. DNS is Data Network Service.

        A-) 1 B-) 2 C-) 1,2 D-) 1,2,3

        """,
    1: """
        Which one is NOT a Http Method ?

        A-) GET B-) POST C-) SEND D-) PUT

        """,
    2: """
        Which are the correct match for HTTP responses status codes ?
        1. 200 = OK
        2. 301 = Moved Permanently
        3. 400 = Not Found
        4. 404 = Bad Request
        5. 500 = Internal Server Error

        A-) 1,3,5 B-) 1,2,5 C-) 1,2 D-) 4,5

        """,
    3: """
        Which one is not joint method for TCP and UDP protocols ?

        A-) socket() B-) bind() C-) close() D-) send()

        """,
    4: """
        Which one is true for PURE P2P architecture ?

        A-) No always-on server
        B-) Arbitrary end systems directly communicate
        C-) Peers are intermittenly connected and change IP address
    """
}
```


D-) Peers are continuously connected

```
        """
    }

answers = {
    0 : "C",
    1 : "C",
    2 : "B",
    3 : "D",
    4 : "D"
}

class Server:

    def __init__(self,serverHost ,serverPort):
        try:
            serverSocket = socket(AF_INET, SOCK_STREAM) #socket() creates socket, AF_INET is IPv4,
SOCK_STREAM is TCP
        except:
            print("Socket cannot be created!!!")
            exit(1)
        print("Socket is created...")
        try:
            serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
        except:
            print("Socket cannot be used!!!")
            exit(1)
        print("Socket is being used...")
        try:
            serverSocket.bind((serverHost, serverPort))
        except:
            print("Binding cannot de done!!!")
            exit(1)
        print("Binding is done...")
        try:
            serverSocket.listen(45)
        except:
            print("Server cannot listen!!!")
            exit(1)
        print("The server is ready to receive")
        while True:
            connectionSocket, addr = serverSocket.accept()
            threading.Thread(target=self.makeQuiz, args=(connectionSocket, addr)).start()

    def makeQuiz(self,client,addr):
        pointsForUser = 0
```

```
questionNum = 0
client.sendto("Please give your username : ".encode(), addr)
name = client.recv(1024)
while True:
    try:
        sended = client.sendto(questions[questionNum].encode(), addr)
        message = client.recv(1024)
        answer = message.decode("utf-8")
        if answer is not "":
            print(name, " says: ", message.decode("utf-8"), " for question : ", questionNum)
            if answer is answers[questionNum]:
                pointsForUser += 5
            else:
                pointsForUser += 0
        except:
            try:
                client.sendto("Points from test: {}".format(pointsForUser).encode(), addr)
                client.close()
                break
            except:
                print(name, " interrupted itself")
                exit(1)
        questionNum += 1

if __name__=="__main__":
    serverPort=12000
    serverHost ='127.0.0.1'
    Server(serverHost,serverPort)
```

Client.py

```
from socket import *
import time

class Client:
    def __init__(self,serverName="127.0.0.1",serverPort=12000,timer=10):
        self.serverName = serverName
        self.serverPort = serverPort
        self.timer = timer
        try:
            self.clientSocket = socket(AF_INET, SOCK_STREAM)
            self.clientSocket.connect((self.serverName, self.serverPort))
        except:
            print("Cannot connect to Server")
            exit(1)

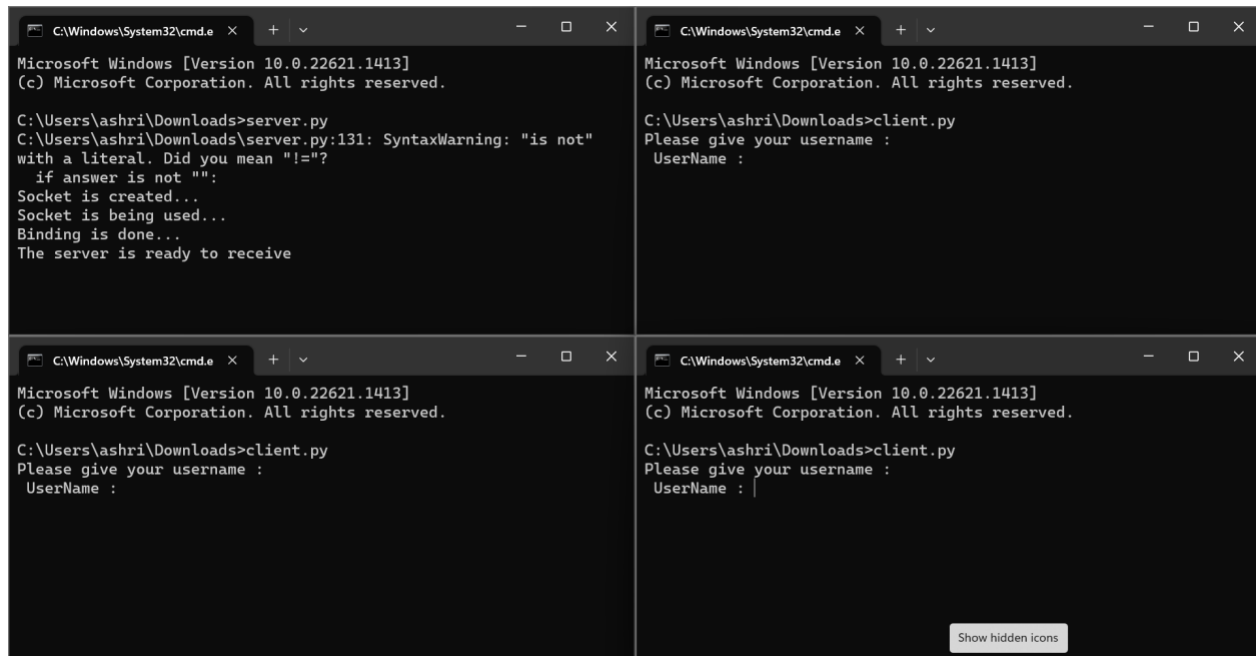
    def GetUsername(self):
        reqName = self.clientSocket.recv(1024)
        print(str(reqName,'utf-8'))
        name = input(" UserName : ")
        self.clientSocket.send(name.encode())

    def MakeQuiz(self):
        comingQ = self.clientSocket.recv(1024)
        print(str(comingQ, 'utf-8'))
        if "Points from test:" in comingQ.decode('utf-8'):
            self.clientSocket.close()
            exit(0)
        before = time.time()
        message = input("Give answer in " + str(self.timer) + " sec: ").upper()
        if time.time() - before > self.timer:
            print(self.timer," seconds passed you got 0 points from this question")
            message = "not possible answer given by user"
        if not message:
            message = "No message"
        self.clientSocket.send(message.encode())
        self.MakeQuiz()

if __name__ == '__main__':
    client = Client()
    client.GetUsername()
    client.MakeQuiz()
```

5. Sample Output

All connections established



```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>server.py
C:\Users\ashri\Downloads\server.py:131: SyntaxWarning: "is not"
with a literal. Did you mean "!="?
  if answer is not "":
Socket is created...
Socket is being used...
Binding is done...
The server is ready to receive

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName :
```

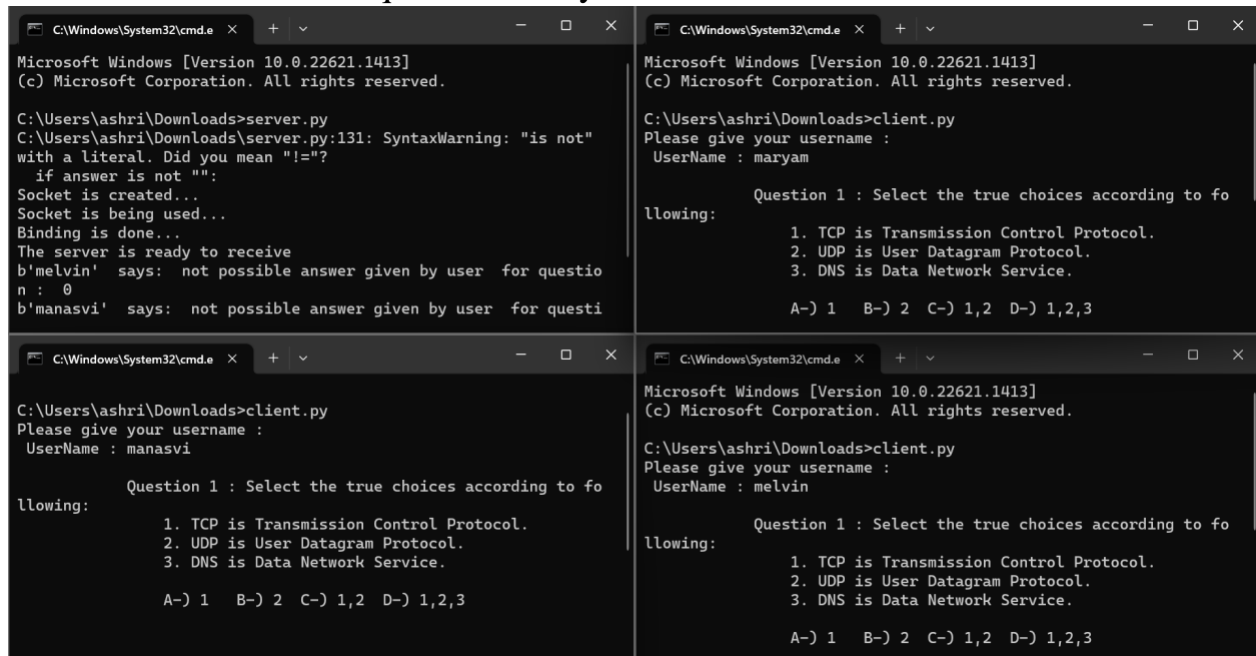
```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName :
```

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName :
```

Each client can take the quiz individually



```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>server.py
C:\Users\ashri\Downloads\server.py:131: SyntaxWarning: "is not"
with a literal. Did you mean "!="?
  if answer is not "":
Socket is created...
Socket is being used...
Binding is done...
The server is ready to receive
b'melvin' says: not possible answer given by user for questio
n : 0
b'manasvi' says: not possible answer given by user for questi
```

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName : maryam

Question 1 : Select the true choices according to fo
llowing:
1. TCP is Transmission Control Protocol.
2. UDP is User Datagram Protocol.
3. DNS is Data Network Service.
A-) 1 B-) 2 C-) 1,2 D-) 1,2,3
```

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName : manasvi

Question 1 : Select the true choices according to fo
llowing:
1. TCP is Transmission Control Protocol.
2. UDP is User Datagram Protocol.
3. DNS is Data Network Service.
A-) 1 B-) 2 C-) 1,2 D-) 1,2,3
```

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>client.py
Please give your username :
UserName : melvin

Question 1 : Select the true choices according to fo
llowing:
1. TCP is Transmission Control Protocol.
2. UDP is User Datagram Protocol.
3. DNS is Data Network Service.
A-) 1 B-) 2 C-) 1,2 D-) 1,2,3
```

Gives 0 points if time limit is crossed

```
Microsoft Windows [Version 10.0.22621.1413]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\ashri\Downloads>client.py
```

```
Please give your username :
```

```
UserName : manasvi
```

```
Question 1 : Select the true choices according to following:
```

1. TCP is Transmission Control Protocol.
2. UDP is User Datagram Protocol.
3. DNS is Data Network Service.

```
A-) 1   B-) 2   C-) 1,2   D-) 1,2,3
```

```
Give answer in 10 sec: B
```

```
10 seconds passed you got 0 points from this question
```

Gives the final score at the end of the test

```
Give answer in 10 sec: A  
10 seconds passed you got 0 points from this question
```

```
Which are the correct match for HTTP responses status codes ?
```

1. 200 = OK
2. 301 = Moved Permanently
3. 400 = Not Found
4. 404 = Bad Request
5. 500 = Internal Server Error

```
A-) 1,3,5   B-) 1,2,5   C-) 1,2   D-) 4,5
```

```
Give answer in 10 sec: B
```

```
Which one is not joint method for TCP and UDP protocols ?
```

```
A-) socket() B-) bind() C-) close() D-) send()
```

```
Give answer in 10 sec: D
```

```
Which one is true for PURE P2P architecture ?
```

- A-) No always-on server
- B-) Arbitrary end systems directly communicate
- C-) Peers are intermittently connected and change IP address
- D-) Peers are continuously connected

```
Give answer in 10 sec: D
```

```
Points from test: 15
```

Server stores all the choices of the users in the database

```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashri\Downloads>server.py
C:\Users\ashri\Downloads\server.py:131: SyntaxWarning: "is not" with a literal. Did you mean "!="?
    if answer is not "":
Socket is created...
Socket is being used...
Binding is done...
The server is ready to receive
b'melvin' says: not possible answer given by user for question : 0
b'manasvi' says: not possible answer given by user for question : 0
b'maryam' says: not possible answer given by user for question : 0
b'maryam' says: not possible answer given by user for question : 1
b'maryam' says: B for question : 2
b'maryam' says: D for question : 3
b'maryam' says: D for question : 4
b'melvin' says: not possible answer given by user for question : 1
b'melvin' says: D for question : 2
b'melvin' says: D for question : 3
b'manasvi' says: not possible answer given by user for question : 1
b'manasvi' says: C for question : 2
b'manasvi' says: C for question : 3
b'manasvi' says: C for question : 4
|
```

6.Conclusion

In conclusion, the online In conclusion, the online quiz application using socket programming is a useful tool for conducting quizzes remotely.

Socket programming is a powerful method for building networked applications. It allows developers to create connections between processes running on different machines, enabling them to communicate and exchange data in real-time.

The use of multithreading allows the server to handle multiple clients simultaneously, making the application scalable and efficient. This project can be extended further by adding features like a timer, user authentication, and a graphical user interface to improve the user experience.

Overall, this project serves as a great example of how socket programming can be used to create powerful networking applications.

7.References

- <https://www.geeksforgeeks.org/socket-programming-cc/>
- <https://www.geeksforgeeks.org/socket-programming-cc/>
- <https://ianfinlayson.net/class/cpsc414/notes/03-sockets>