Q1 – Design a network application protocol for this application and justify your design decision. In your protocol design, you should address the following.

1. The communication patterns of the network application.

- Our communication between client and server will be simple. Once a connection is established between client and server, the server will communicate with the client using our communication rules and exchange information. Once either side needs to close the connection, the other side will do so as well. This communication will be done in a separate thread for each client on the server, which will allow multiple clients to connect to the server.

2. Define the protocol design goals.

- The protocol for our application will be the default that is set through python sockets on the TCP/IP setting. This is a simple transfer of text through the socket's stream back and forth from client and server where one waits with the recv() function and one sends with the sendall() function.

3. Define the message format, structure and semantics

- The message format will be encoded strings using python's encode() function, and decode() upon receiving. Each message will be a simple all-upper-case string, except for when sending jobs. The job titles will not be all upper case. For example – Job = "Craft and send TCP packet" and any other message = "YES", "NO", "REJECT" etc.

4. Design the communication rules.

- Our communication rules will follow TCP/IP. Once connected through TCP, server will send client job in form of text. Client will either decline and send back a string saying that the job was rejected or accept the job and send back string containing information of completed job. After job is completed, server will ask the client if they want another job. The client will respond back "YES" if they do, and "NO" if they don't. The server then terminates connection upon receiving "NO" or loops the job process upon receiving "YES". Once either the client or server terminates connection, the other will do so promptly as well.

Q2 – Argue the need for a new application layer protocol for this network application instead of using existing standard protocols (e.g. HTTP, SMTP, WebSocket, etc.)

- The reason that we would need a new application layer for this program rather than using standard protocols is because of the complexity of the program itself. The reason we cannot use HTTP is because our program does not require communication between web browser and web servers. The next one would be SMTP; SMTP is used for mail transfer and is used for emails and fetching emails from servers and our program is not related to email and mail transfer. The reason that a WebSocket would not work would be that our program itself does not have any correlation regarding browsers and working with the browser. For those reason we would need a new application layer to support the program that we are building which is to create a network between an employer and people who are looking for work currently.

Q3 – Provide the implication source code of your network application protocol with sufficient test cases based on the design goals, message philosophy (format, structure, semantics), and Communication rule.

GitHub repository: https://github.com/abdulazizkhan75/A2_CompNetworks

Test cases: As you can see, multiple clients connect to one server and perform tasks simultaneously. Once a connection is closed on one side, it is closed on the other side.

```
How many clients? >2
Created server localhost on port 5555
Server: Waiting for new connection...
Client 1: Attempting to connect to localhost
Client 1: Successfully connected to server
Client 2: Attempting to connect to localhost
Client 2: Successfully connected to server
Server: Connected to client #1: ('127.0.0.1', 57498)
Server: Total clients:  1
Server: Asking client 1 to do job - Craft and send TCP packet
Server: Waiting for new connection...
Server: Connected to client #2: ('127.0.0.1', 57500)
Cleint 1: Recieved job - Craft and send TCP packet
Server: Total clients:  2
Client 1: COMPLETED Craft and send TCP packet
Server: Client 1 successfully completed job - Craft and send TCP packet
Server: Asking client 1 to perform job again...
Server: Asking client 2 to do job - Craft and send TCP packet
Server: Waiting for new connection...
Client 1: Server asked to do another job
Client 1: Response - NO
Client 1: Closing connection with server
Cleint 2: Recieved job - Craft and send TCP packet
Client 2: COMPLETED Craft and send TCP packet
Server: Client 2 successfully completed job - Craft and send TCP packet
Server: Asking client 2 to perform job again...
Client 2: Server asked to do another job
Client 2: Response - NO
Client 2: Closing connection with server
Server: Client 2's response to do job again: NO
Server: Closing connection with client  2
Server: Client 1's response to do job again: NO
Server: Closing connection with client  1
```