Part 2: Design and Programming Assignment

Q1)

1) We are using a client server model as the job creator will be server and the client will be the job-seeker. The job seeker will be seeking jobs and the job creators will be offering the jobs. We are using push communication since the job creator is sending out jobs and the job seeker will complete the job in a given amount of time and will notify the creator when the job is complete, not necessarily giving an immediate response to the server. We are using one2one communication since a job creator can only send jobs to a job seeker one job at one time, although multiple jobs can be assigned to multiple different job seekers.

2) Simplicity, efficiency, scalability are the main qualities that we want to prioritize for our project.

   Do we need a reliable message exchange? **YES because we were instructed to use TCP which promoted reliability.**
   Is communication security important? **NO because the case does not specify the need for authentication or authorization of job seekers.**
   Do we need persistent connections? **YES because the connection could be terminated or maintained depending on the job**
   Do we have any bandwidth or latency restrictions or requirements? **NO**
   How do I handle error and failure? **We are using TCP connection and TCP will handle packet loss or corruption.**
   What are your scalability requirements? **The case does not specify any scalability requirements, and also specified to be as simple as possible.**

3) In general protocols could be either text-protocols, or binary-protocols. We will be using text protocol as text is simpler than using binary communication and more akin to the higher-level protocols we are using. Text protocols are also easier to understand, design, extend, test, and debug. Our seeker message format/semantic is the following:

```
#   Switch case for different message types
    if messageType == 1:
      #   Services message
      message =   {'header':messageType,
                   'services':services}
    elif messageType == 2:
      #   Accept message
      message =   {'header':messageType,
                   'Accept':accept}
    else:
      #   Completion message
```

```
message =  {'header':messageType,
            'job':job,
            'status':status,
            'result':result}
```

The creator structure will be the following:

```
#   Switch case for different message types
    if messageType == 1:
      #   Job message
      message =   {'header':messageType,
            'job':job}
    else:
      #   Acknowledge message
      message =   {'header':messageType,
            'acknowledge':acknowledge}
```

Services message is a data message that the seeker sends to the creator, letting it know which services its offering. **Accept message** is a data message that the seeker sends to the creator, letting the creator know if the job is accepted or rejected. **Completion message** is a data message that the seeker sends to the creator, letting it know whether the seeker has completed the job and the result of the job. **Job message** is a data message that the creator sends to the seeker, letting the seeker know what job it has been assigned. **Acknowledge message** is a message that the creator sends to the seeker, letting the seeker know that is acknowledges whether the seeker accepted or rejected a job?

4) The job seeker (client) will first connect to the job creator (server) via TCP. Once the connection has been established the job seeker will send a message to the creator containing the services it can provide. When the creator receives this message, it will align the seeker's services with a job and will send a message back to the seeker if a job is available. The seeker, upon receiving the job offer, will then decide if they want to accept or reject the job and will send a message back to the creator about their decision. The creator will then send back a message to the seeker acknowledging that they know the seeker accepted or rejected the job. The job seeker upon completing their job would then send back a data message back to the creator about the status and result of their job.

Q2) Our protocol is specially designed for the job seekers and job creators' task, it only contains the information needed for the seeker and creator to communicate and complete jobs, which streamlines the communication between creator and seeker. This makes our protocol specialized for the task that has been given to us, as it eliminates any extra semantics making the communication as frictionless as possible, and thus much faster than traditional protocols.

Q3) Github link: https://github.com/Cosy8/Communication_Protocol