1. Based on what you've read so far, would you define this application as a distributed system? Why or why not? Use the design goals listed in Ch 1 to justify your position.
   1. The book defines a distributed system as: “A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.” The client and server both are autonomous computing elements but they combine to a user to seem as one system.
   2. The design goals are
      1. Supporting resource sharing – this is part of the system because clients can access resources (functionality/information) stored by the server.
      2. Making distribution transparent – the book talks about differing degrees being useful in other cases

|  |  |
| --- | --- |
| **Transparency** | **As applied** |
| Access | The data is retrieved by entering a command. There is a transformation to send this data and get back what the user wants but a casual user would not know that. |
| Location | The server could be on a different machine or a different window and still function similarly. |
| Relocation | As long as connection can be maintained, (internet or otherwise) a client user could not tell if a server was moving. |
| Migration | The server can easily move to another location and, at most, the client would just be entering a different ip/port. |
| Replication | The server is not currently replicated but it is on different systems and the client would not know that. |
| Concurrency | There can be several clients and any given client would not know that. |
| Failure | If the server crashes, the client might get a message about there connection ending but other than that they likely would not know. |

* + 1. Being Open
       1. Interoperability – all pieces were made by the same “manufacturer” in this case but it could be the case that one group wrote server code for a TCP server and another group for the client and they work together.
       2. Portability – the TCP server should be able to work with a different client that connects to TCP servers if they were set up to send useful information.
       3. Extensibility – It would be easy to add more functionality to the server as it is set up.
    2. Separating policy from Mechanism: The project is setup into two relatively small parts that could be replaced. It could be made even smaller by spreading out more of the functionality but that is an appropriate scale for this project.
    3. Being scalable –
       1. Size scalability: if the server was put on a powerful computer, it could scale up and change how many connections it would allow.
       2. Geographical scalability: system requires a connection which should scale with geography
       3. Administrative scalability: current system model can be easily managed through modifying the server
  1. Because of all of these factors, I would say the client/server system is a distributed system.

1. Based on your Chapter 2 reading, what architectural style(s) does your software leverage? Explain why it fits.
   1. List of architecture styles
      1. Layered architectures
      2. Object-based architectures
      3. Resource-centered architectures
      4. Event-based architectures
   2. This project used the layered architecture because there was the application layer that the clients use and then the processing layer that processes the commands which is the server. It also uses an object-based architecture. There are at least two objects in this implementation, the client and the server. Some designs will have more objects. It does not really follow the resource-centered as currently constructed. The event-based architecture seems to be similar to a publish-subscribe framework and this implementation doesn’t really use this.
2. Based on your Chapter 2 reading, what system architecture does your software leverage?
   1. It follows the centralized organization, simple client-server architecture. We have a client and a server. The server provides a specific service (providing information) and the client requests that service.
3. What steps would you need to take to "evolve" your code into the following types of systems. List at least three major tasks each.
   1. A three-tiered architecture pulling data from an SQL database.
      1. Create a useful database (thus making it 3 tiers)
      2. Add a way for the server to interface with the database and for the database to interface with the server
      3. Find a way to keep track of the outgoing connections on the server on both ends to make sure information is not lost or exposed incorrectly
   2. A node of a peer-to-peer system (structured or unstructured)
      1. Set every process so it can function as both a client and a service
      2. Set up appropriate communication channels with overlay
      3. Spread out the information appropriately among the parts of the system
   3. An edge-server system
      1. Add more security as it is transmitting over the internet
      2. Make sure it is compatible with ISPs and enterprise network specifications
      3. Apply filtering and transcoding functions as necessary
      4. Set an origin server for the content to originate from