John Beighle's 689 HW1 Questions and Answers 1/17/2020

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.

Yes, of course. It demonstrates both key characteristic: "collection of autonomous computing elements" and it appears to clients as a "single coherent system". It also makes use of other key DS concepts like: "resource sharing" by sharing both computational resource loads and data, "
making distribution transparent" the system appears a single coherent interface to users, "
"interoperability, composability, and extensibility" it uses standard, open, computer architecture interfaces and transports, e.g. sockets, TCP/IP, and can be scaled to a broad array of client types. It is built to bolt on scalability easily in terms of size and geography, the client can be administered by one group and the server a different group. It is positioned to scale both up and out easily being programmed in C++ which has extensive capabilities both in terms of inherent native libraries but also almost unlimited 3rd party options.

2. Based on your Chapter 2 reading, what architectural style(s) does your software leverage? Explain why it fits.

Client server is by nature a layered architecture. It is also implements event based coordination and thus closely resembles a pub-sub style. It is not direct in that the event structure is determined after randomly connecting to the server service. Events are generated on the client and passed on the event bus. The events albeit, in terms of software sophistication, are rudimentary.

3. Based on your Chapter 2 reading, what system architecture does your software leverage?

Basic two-tiered client server architecture that maintains an open socket, stateful, connection to transfer events, messages back and forth.

4. What steps would you need to take to "evolve" your code into the following types of systems. List at least three major tasks each:

A. A three-tiered architecture pulling data from an SQL database.

1. Build a useful data structure
2. Consider building object wrappers for data structures
3. Consider resource requirements such as connection pools, load balancing

B. A node of a peer-to-peer system (structured or unstructured).

1. Learning the peer-to-peer architecture and API
2. Providing highly reliable, fault-tolerant connection pools
3. Implementing appropriate load and bandwidth sufficient scalability measures

C. An edge-server system

1. Implementation of advanced security protocols and measure
2. Implementation of multi-factor client authentication
3. Considering 3rd party COTS solutions and having a vendor chili cookoff