KEVIN JOHNSON

MODULE 1 READING COMPREHENSION

1. **Define what Reliability, Scalability, and Maintainability mean in terms of Data Systems.**

   Reliability ensures the system should continue to work correctly (perform the correct function at the desired level of performance) even in the face of adversity (hardware or software faults and human error). Typical expectations for reliability include:

   - The application performs the function that the user expected.
   - It can tolerate the user making mistakes or using the software in unexpected ways.
   - Its performance is good enough for the required use case, under the expected load and data volume.
   - The system prevents any unauthorized access and abuse.

   Scalability means as the system grows (in data volume, traffic volume, or complexity), there should be reasonable ways of dealing with that growth.

   Maintainability refers to over time, many different people will work on the system and should be able to productively.

2. **Describe three types of issues that cause problems with Reliability.**

   The below three issues cause problems with reliability:

   - Hardware Faults: Hard disk crashes, RAM becoming fault, a blackout of the power grid, and someone unplugging the wrong network cable, are examples of hardware faults. Recently, there has been a move toward systems that can tolerate the loss of entire machines, by using software fault-tolerance techniques in preference to or in addition to hardware redundancy.

   - Software Errors: Systematic errors within the system are harder to anticipate. Because they are correlated across nodes, they tend to cause many more system failures than uncorrelated hardware faults. Below are a few examples:
     1. A software bug that causes every instance of an application server to crash when given a particular bad input.
     2. A runaway process that uses up some shared resource such as CPU time, memory, disk space, or network bandwidth.
     3. A service that the system depends on that slows down, becomes unresponsive, or starts returning corrupted responses.
     4. Cascading failures, where a small fault in one component triggers a fault in another component, which in turn triggers further faults.

- Human Errors: The best systems combine several approaches to deal with human errors:
    1. Design systems in a way that minimizes opportunities for error.
    2. Decouple the places where people make the most mistakes from the places where they can cause failures.
    3. Test thoroughly at all environments, from unit tests to whole-system integration tests and manual tests.
    4. Allow quick and easy recovery from human errors, to minimize the impact int eh case of a failure.
    5. Set up detailed and clear monitoring, such as performance metrics and error rates.
    6. Implement good management practices and training.

3. Describe a few ways we can deal with and plan for Scalability.

Some approaches for dealing with load include scaling up (moving to a more powerful machine) and scaling out (distributing the load across multiple smaller machines). Some systems are elastic, where they can automatically add computing resources when they detect a load increase.

Dealing with load is often specific to the application. First, load parameters should be defined, which will depend on the architecture of the system. Some examples of parameters include requests per second to a web server, the ratio of reads to writes in a database, the number of simultaneously active users in a chat room, the hit rate on a cache, etc. Once load has been described for the system, you can investigate what happens when the load increases. Below are two ways:

- When you increase the load parameter and keep the system resources unchanged, how is the performance of your system affected?
- When you increase a load parameter, how much do you need to increase the resources if you want to keep performance unchanged?

4. Describe the design principles to take into consideration when dealing with Maintainability.

There are three key design principles to take into consideration when dealing with maintainability:

- Operability: Make it easy operations teams to keep the system running smoothly.

- Simplicity: Make it easy for new engineers to understand the system, by removing as much complexity as possible from the system.

- Evolvability: Make it easy for engineers to make changes to the system in the future by adapting it for unanticipated use cases as requirements change.