**Scalability:**

The ability for the system to perform and operate as the number of users or requests increases. Scalability is achievable with Horizontal or vertical scaling of the machine or simply attaching AutoScalingGroup.

**Traffic Pattern**: Understand the traffic pattern of the system. It’s not cost-efficient to spawn as many machines as possible, even if it is underutilisation.

* **Diurnal Pattern**: Traffic increases in the morning and decreases in the evening for a particular region.
* **Global / Regional**: Regional Heavy usage of the application.
* **Thundering Herd**: Many users are requesting resources, but only a few machines are available to serve the burst of traffic. These could occur during **peak time** or in **densely populated areas**.

**Elasticity**: Ability to quickly spawn a few machines to handle the burst of traffic and gracefully shrink when the demand is reducing.

**Latency**: Ability to serve the request as quickly as possible. This also includes optimising the algorithms and replicating the system near users to reduce the round trip of a request. An average google search takes 400 ms.

**Availability**

It is measured as a percentage of uptime and defines the proportion of time that a system is functional and working. Availability is affected by system errors, infrastructure problems, malicious attacks and system load.

**Deployment Stamps**: Deploy multiple independent copies of application components, including data stores

**Geodes**: Deploy backend services into a set of geographical nodes, each of which can service any client request in any region.

**Extensibility**

Extensibility measures the ability to extend a system and the effort required to implement the extension. The extension can be through adding new functionality or modifying existing functionality. The principle provides for enhancements without impairing current system functions.

**Modular / Reusability**: Reusability, together with extensibility, allows technology to be transferred to another project with less development and maintenance time, as well as enhanced reliability and consistency.

**Pluggability:**Ability to easily plug other components like in the case of microkernel architecture.

**Consistency**

Consistency guarantees that every read returns the most recent write. This means that after the execution of every operation, the data is consistent across all the nodes, and thus all clients see the same data at the same time, no matter which node they connect to. Consistency improves the **data freshness**.

**Resiliency**

A system can gracefully handle and recover from accidental and malicious failures. Detecting failures and recovering quickly and efficiently is necessary to maintain resiliency.

**Recoverability**: The preparatory processes and functionality enable you to return your services to an initial functioning state after an unintended change. Unintended changes include the soft or hard deletion or misconfiguration of applications.

* **Disaster Recovery**: Disaster recovery (DR) consists of best practices designed to prevent or minimise data loss and business disruption resulting from catastrophic events — everything from equipment failures and localised power outages to cyberattacks, civil emergencies, criminal or military attacks, and natural disasters.

**Design Patterns**:

* **Bulkhead**: Isolate elements of an application into pools so that if one fails, the others will continue to function.
* **Circuit Breaker**: Handle faults that might take a variable amount of time to fix when connecting to a remote service or resource.
* **Leader Election**: Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.

**Usability**

Usability can be described as the capacity of a system to provide a condition for its users to perform the tasks safely, effectively, and efficiently while enjoying the experience. It is the degree to which specified consumers can use software to achieve quantified objectives with effectiveness, efficiency and satisfaction in a quantified context of use.

**Accessibility**: Make the software available to people with the broadest range of characteristics and capabilities. This includes users with deaf, blind, colourblindness, etc.

**Learnability**: How easy users can learn how to use the software?

**API Contract**: For Internal teams, understanding the API Contracts can help easily pluggable into any system.

**Observability**

Observability is the ability to collect data about program execution, internal states of modules, and communication between components. To improve observability, use various logging and tracing techniques and tools.

**Logging**: There are different types of logs generated within each request: event logs, transaction logs, message logs and server logs.

**Alerts & Monitoring**: Prepare monitoring dashboards, create SLI (Service Level Indicators) and set up critical alerts.

**L1 / L2 / L3**: Setup on-call support process for L1 / L2. L1 support includes interacting with customers. L2 support manages the tickets routed to them by L1 and helps troubleshoot. L3 is the last line of support and usually comprises a development team which addresses the technical issues.

**Security**

Degree the software protects information and data so that people or other products or systems have the degree of data access appropriate to their types and levels of authorisation. This family of characteristics includes *confidentiality* (data is accessible only to those authorised to have access), *integrity* (the software prevents unauthorised access to or modification of software or information), *nonrepudiation* (can actions or events be proven to have taken place), *accountability* (can user actions of a user be traced), and *authenticity* (verifying the identity of a user).

**Auditability**: Audit trails track system activity so that when a security breach occurs, the mechanism and extent of the breach can be determined. Storing audit trails remotely, where they can only be appended, can keep intruders from covering their tracks.

**Legality**:

* **Compliance**: Adhere to data protection laws like GDPR, CCPA, SOC2, PIPL or FedRamp.
* **Privacy**: Ability to hide transactions from internal company employees (encrypted transactions so even DBAs and network architects cannot see them).

**Authentication**: Security requirements to ensure users are who they say they are.

**Authorisation**: Security requirements to ensure users can access only certain functions within the application (by use case, subsystem, webpage, business rule, field level, etc.).

**Durability**

Durability is the solution ability of software’s serviceability and meet users’ needs for a relatively long time.

**Replication**: involves sharing information to ensure consistency between redundant resources to improve reliability, fault-tolerance, or accessibility.

**Fault Tolerance**: It is the property that enables a system to continue operating correctly in the event of the failure of one or more faults within some of its components.

**Archivability**: Will the data need to be archived or deleted after some time? (For example, customer accounts will be deleted after three months or marked as obsolete and archived in a secondary database for future access.)

**Agility**

It has become today’s buzzword when describing a contemporary software method. An associate agile team could be a handy team able to befitting reply to changes. Modification is what software development is highly abundant.

**Maintainability**: How easy it is to apply changes and enhance the system? — Represents the degree of effectiveness and efficiency to which developers can modify the software to improve it, correct it, or adapt it to changes in environment and requirements.

* **Testability**: how easily developers and others can test the software
* **Ease of development**: the degree to which developers can modify the software without introducing defects or degrading existing product quality

**Deployability**: The time to get code into production after committing the deployment time.

* **Installability**: Ease of system installation on all necessary platforms.
* **Upgradeability**: Ability to easily/quickly upgrade from a previous version of this application/solution to a newer version on servers and clients.
* **Portability**: Does the system need to run on more than one platform? (For example, does the frontend need to run against Oracle and SAP DB?

**Configurability**: The end users can easily change aspects of the software’s configuration (through usable interfaces).

* **Compatibility**: Degree to which a product, system, or component can exchange information with other products, designs, or members and perform its required functions while sharing the same hardware or software environment.

**Conclusion**

Now, that you are familiar with the NFRs or Architectural Characteristics, you may be thinking that which one will fit your project needs. Or may be all of them are required in every project. So how to proceed with adopting these characteristics within your project?

Once you get the functional requirement, try to find the bottleneck in the system which may add obstacle to primary function. And how to find the bottleneck? Try answering few of such questions:

* Will the system perform in 100M / 1B userbase?
* Will my system handles 10,000 concurrent request?
* Am I handling the data in secured way?
* Can I add more features easily without impacting the existing working features?
* and many more…

Some of these questions can help to identify the bottleneck or less performing area which are potential start point to improve the overall reliability of the system.