**Faults and Failures**

Whenever a system does not perform as per the desired purpose.

Fault is the cause and Failure is the effect of non-functioning.

Whenever you design a system, it is going to fail.

It’s our job to anticipate possible faults and failures and desi a fault-tolerant or fail-safe system.

Examples of Faults:

Network fault, network timeout, hardware issues, machine crashes, system out of memory, system running on very high CPU, human errors – software bugs.

These faults can cause huge loss to businesses.

Hence we must:

1. Understand type of faults.
2. Tolerate faults.
3. Make system fail-safe.

Suppose, a network line goes down, and our app on server is not able to serve requests.

In such **hardware** **issues**, we can have **multiple instances** of the application, so that even if one line goes down, other instances can serve the requests, hence system is now fault-tolerant.

Now what if, you have software **bugs**, even if you keep multiple instances, it will lead to failure on certain actions. Hence to avoid software bugs, we do thorough **testing** of bugs, write test cases, but still there might be some test cases unhandled or actions never thought of. In that cases, if system fails, the frontend must show a **user-friendly message** to user and handle these faults gracefully.

Faults can occur in **Databases** too, in that case as well having **multiple instances** of DB helps. There are certain **fail-over mechanisms**, where when one DB goes down, other DB instance takes over and perform reads and writes.

Types of faults:

Faults can generally be hardware issues or network issues, software bugs, natural calamity like earthquakes, etc.

Faults can also be categorized as:

**Transient Fault**

1. Occur of a short duration of time.
2. Hard to locate.

**Permanent fault**

1. Continues to occur until fixed.
2. Easy to identify.

Real-Life example.

In airplanes, we have 4 engines.

If one engine fails, we have 3 more as backups – this is **fault tolerance.**

If still plane will be going down, we have emergency-exits, safety jackets and parachutes – this is called, making system **fail safe** or failing gracefully (where certain faults cannot be tolerated).

Using same mindset, we must build systems.