**Seven Principles of Software Testing**

**1. Testing shows the presence of bugs**

* Testing cannot prove that the application is defect free
* Testing reveals the defects in the Software, which doesn't mean that the application will become defect free because of testing.
* ie The goal of the testing is to find as many defects as possible instead of making the application defect free

**Goal:** Identify as many defects as possible, not to guarantee bug-free software.

**2. Exhaustive testing is impossible**

* Testing with all valid and invalid combinations of input may not be possible
* Example: Testing an input field which can intake numbers from 1 to 1 Lack
* We test them based on risk and priorities

**Approach:** Focus on **risk-based** and **priority-based** testing.

**3. Early Testing**

* Testing should start from the early stages of SDLC
* Defects found in early stages of the SDLC, are cheaper to fix than those found in the later stages
* Defects identified during earlier stages cost less
* Defects identified after release are expensive

**Why:** Detecting and fixing defects early is **cheaper** and **less time-consuming** than doing it later.

**4. Testing is Context Dependent**

* Testing of different domains has to be done differently
* The risk associated with different application domains in different
* Example: Testing Banking application is different for Schooling Assignments application

**Example:** Safety-critical systems (e.g., medical or aerospace) need **rigorous testing**, while e-commerce apps may need **more usability and performance testing**.

**5. Defect Clustering**

* The majority of the defects come from a small number of modules
* The defects won't be equally distributed across the application

**Tip:** Focus testing efforts more on **high-risk** or **historically buggy areas**.

**6. Pesticide Paradox**

* Pesticide Paradox principle says that if the same set of test cases are executed again and again over the period of time, these tests won't capable enough to identify new defects in the Software
* In order to overcome Pesticide Paradox, we need to review and update the test cases regularly

**Solution:** Regularly **review and revise** test cases, and add **new** and **different** ones

**7. Absence of Error - fallacy**

* 99% of bug-free software may still be unusable, if wrong requirements were incorporated into the software and the software is not addressing the business needs

**Reminder**: Software must also meet **business** and **user requirements**—not just be error-free.