**Unit Testing:**-

**What?**

* Unit testing is nothing but testing of each unit or an individual component of the application.
* It is the 1st level of functional testing.
* The purpose of unit testing is to test the correctness of isolated code.
* A unit component is an individual function or code of the application.
* Unit testing comes under White Box Testing and usually done by the Developers.

**When?**

* Whenever the application is ready and given to the Test Engineer, then will start checking every component of the module one by one and this process is known as Unit Testing.

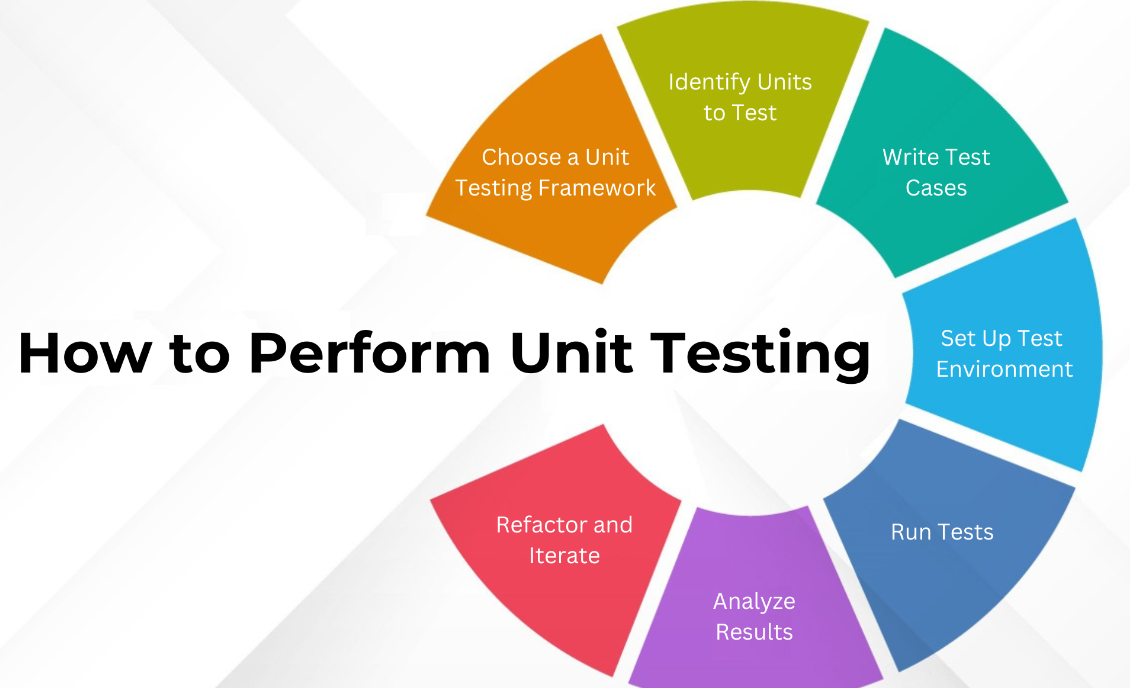
**Why?**

**Early Error Detection:**

* Identifies bugs and issues at an early stage, which is easier to fix.
* Unit Testing helps in the documentation.

**Saves Time and Cost:**

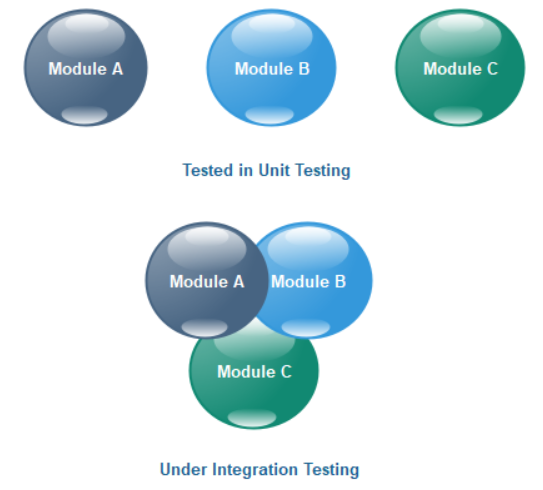
* Detecting and fixing bugs early is much more cost-effective than addressing them after the s/w is fully developed.

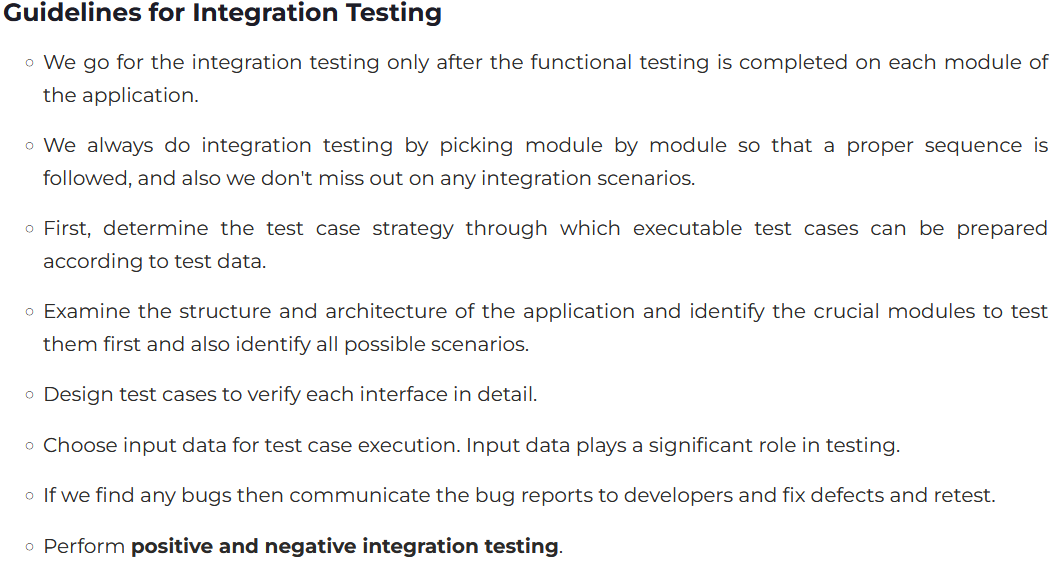


**Integration Testing:-**

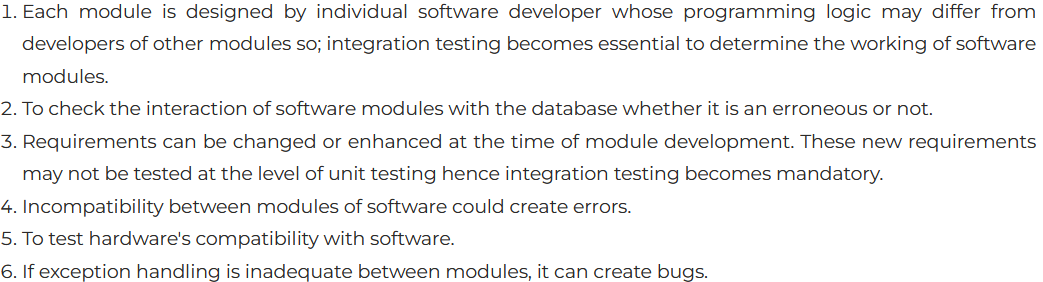
**What?**

* Integration testing is the second level of the software testing process comes after Unit Testing.
* In Integration Testing, units or individual components of the software are tested in a group.
* The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.
* It focuses on verifying the interactions and data flow between integrated modules to ensure that they work together as expected.
* The goal of integration testing is to check the correctness of communication among all the modules.
* Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing****.**

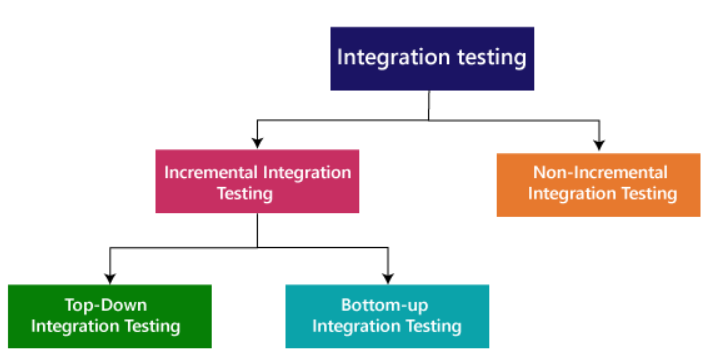


* 

**Reason Behind Integration Testing**



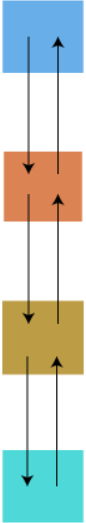
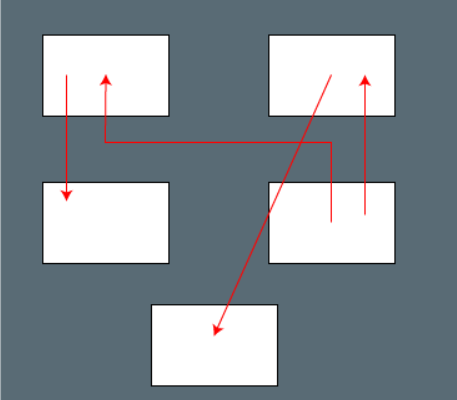
**Types of Integration Testing**



* **Incremental integration testing**
  + In the Incremental Approach, modules are added in ascending order one by one or according to need.
  + In this type of testing, there is a strong relationship between the dependent modules.
  + Suppose we take two or more modules and verify that the data flow between them is working fine. If it is, then add more modules and test again

**For example:** Suppose we have a Flipkart application, we will perform incremental integration testing, and the flow of the application would like this:

Flipkart→ Login→ Home → Search→ Add cart→Payment → Logout

(I**ncremental**) (**Non-incremental**)

* **Non-incremental integration testing**
  + We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the **Big bang method**.

**Approach**:

* **Top-Down Testing**: Starts with testing high-level modules and progressively integrating and testing lower-level modules.
* **Bottom-Up Testing**: Begins with lower-level modules and moves up the hierarchy.
* **Big Bang Testing**: All modules are combined and tested simultaneously. This approach can be challenging since isolating failures is complex.
* **Hybrid Testing (Sandwich)**: Combines Top-Down and Bottom-Up approaches.

**Stubs and Drivers**:

* **Stubs**: Used as temporary replacements for lower-level modules in top-down integration.
* **Drivers**: Act as temporary higher-level modules in bottom-up integration to test lower-level modules.

**Examples of Integration Tests**:

* Testing interaction between a login module and a dashboard module.
* Verifying that data flows correctly between a front-end application and an API.

