1. What is integration testing ?

This Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.

Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

This testing is to expose faults interaction between integrated units.

Test drivers and test stubs are used to assist in Integration Testing.

Integration testing is done by a specific integration tester or test team.

Integration testing tests integration or interfaces between components, interactions to

different parts of the system such as an operating system, file system and hardware or

interfaces between systems.

Components may be code modules, operating systems, hardware and even complete systems.

There are 2 levels of Integration Testing

1. Component Integration Testing
2. System Integration Testing
3. Mention what bing bang testing is ?

In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.

Big Bang testing has the advantage that everything is finished before integration testing starts.

The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

Here all component are integrated together at **once**, and then tested.

. **Advantages:**

 Convenient for small systems.

.**Disadvantages:**

Fault Localization is difficult.

Given the sheer number of interfaces that need to be tested in this approach, some

interfaces links to be tested could be missed easily.

Since the integration testing can commence only after “all” the modules are

designed, testing team will have less time for execution in the testing phase.

Since all modules are tested at once, high risk critical modules are not isolated &

tested on priority. Peripheral modules which deal with user interfaces are also not

isolated and tested on priority.

1. What is purpose of exit Criteria ?

Exit criteria in a test plan define the conditions that must be met before a testing phase can be considered complete and the software is ready for the next stage. They ensure that sufficient testing has been conducted requirements are met, and the software meets quality standards. Essentially, they serve as a "green light" for moving forward in the software development lifecycle.

1. What is functional system testing ?

**A requirement that specifies a function that a system or system component must perform.**

Requirement may exist as a text document and/or a model.

There is 2 type of it

1. Requirement Based Functional Testing
2. Process Based Testing

**Accuracy :** Provision of right or agreed results or effects

**Interoperability:** Ability to interact with specified systems

**Compliance :** Adhere to applicable standards, conventions, regulations or laws

**Auditability :** Ability to provide adequate and accurate audit data

**Suitability :** Presence and appropriateness of functions for specified tasks

1. **Requirement Based Testing**

Testing against requirements and specifications

Test procedures and cases derived from

detailed user requirements

system requirements functional specification

User documentation/instructions

high level System design

Starts by using the most appropriate black-box testing techniques

May support this with white-box techniques (e.g. menu structures, web page navigation)

Risk based approach

1. **Business Process Based Testing**

Test procedures and cases derived from:

Expected user profiles

Business scenarios

Use cases

Testing should reflect the business environment and processes in which the system will operate.

Therefore, test cases should be based on real business processes.

Business Process Testing (BPT) is a software testing methodology that focuses on validating end-to-end business processes within an organization’s software applications.

1. Difference between QA V/S QC V/S Tester .

Quality Assurance (QA) and Quality control (QC) are both important methods in software engineering to get high-quality software.

Quality Assurance (QA) prevents software defects or minimizes the number of defects in software before delivery by making sure that proper methods and processes are followed during the software development process.

Whereas Quality Control (QC) identifies and fixes the defects or errors that exist after development.

1. Difference between smoke & sanity .

**Smoke testing** and **Sanity testing** are two important types of testing in software development.

Both the tests are used to validate the functionality of a software product and identify any critical issues, however they are quite different in their scope.

The basic difference between the two is that smoke testing is one that ensures that the features of an application are working fine or not, whereas sanity

testing is one that is performed to test whether all the errors have been fixed after the build of the application.

|  |  |  |
| --- | --- | --- |
| **No.** | **Smoke Testing** | **Sanity Testing** |
| 1. | It is done to ensure that the functionalities of the program are working fine. | It is done to check if the bugs have been fixed after the build or not. |
| 2. | It is considered a subset of acceptance testing. | It is considered as a subset of regression testing. |
| 3. | It is documented. | It isn't documented. |
| 4. | It can be done by developers or testers. | It is generally performed by testers. |
| 5. | It may or may not be stable. | It is stable. |
| 6. | It is scripted. | It is not scripted. |
| 7. | It is done to understand the stability of the system or product. | It is done to understand the measure of rationality of the product. |
| 8. | It is done to test the functionality of the product or system. | It is used only in case of modified or defective functions of product. |
| 9. | It can be done manually or using automation. | It is generally done manually, not using automation. |
| 10. | It is done when a new product is built. | It is done after completing regression testing. |

1. 7 keys principal of testing .
2. Testing shows presence of defects
3. Exhaustive testing is possible
4. Early testing
5. Defect clustering
6. Pesticide paradox
7. Testing is context dependent
8. Absence of error fallacy
9. Testing shows presence of defects

Testing can show that defects are present, but cannot prove that there are no defects in software

1. Exhaustive testing is possible

We cant test every test cases ,if you try to test all the test cases again & again then you will waste your time & money.

That is we must Prioritise our testing effort using a Risk Based Approach.

1. Early testing

Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

Testing activities should start as **early** as possible in the development life cycle.

These activities should be focused on defined objectives – outlined in the Test Strategy.

1. Defect clustering

A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.

Defects are not evenly spread in a system They are clustered In other words, most defects found during testing are usually confined to a small number of modules.

Similarly, most operational failures of a system are usually confined to a small number of modules.

1. Pesticide paradox

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.

To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

1. Testing is context dependent

Different software needs different type of testing .

Like e commerce web site is different from android application.

So it depend on context of software.

1. Absence of error fallacy.

If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help .

If we build a system and, in doing so, find and fix defects.

It doesn’t make it a **good** system, even after defects have been resolved it may still be unusable and/or does not fulfil the users.

1. What is Error, Defect, Bug and failure?

Error : A mistake in coding its called error. Developer uses this word most.

Defect: error found by tester is called defect.

Failure : Defect accepted by development team then its called bug, builds does not meet the requirements then it is called failure.