2. INTEGRATION TESTING AND SYSTEM TESTING

Integration testing

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed.

Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application.

The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

Integration testing can be done by picking module by module. This can be done so that there should be a proper sequence to be followed. And also if you don't want to miss out on any integration scenarios then you have to follow the proper sequence.

Exposing the defects is the major focus of the integration testing and the time of interaction between the integrated units.

Integration test approaches – There are four types of integration testing approaches. Those approaches are the following:

1. Big-Bang Integration Testing – It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module

Testing. In simple words, all the modules of the system are simply put together and tested.

This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during Big Bang integration testing is very expensive to fix.

Big-bang integration testing is a software testing approach in which all components or modules of a software application are combined andtested at once.

This approach is typically used when the software components have a low degree of interdependence or when there are constraints in the development environment that prevent testing individual components.

The goal of big-bang integration testing is to verify the overall functionality of the system and to identify anyintegration problems that arise when the components are combined.

While big-bang integration testing can be useful in some situations, it can also be a high-risk approach, as the complexity of the system and the number of interactions between components can make it difficult to identify and diagnose problems.

Advantages:

1. It is convenient for small systems.
2. Simple and straightforward approach.
3. Can be completed quickly.
4. Does not require a lot of planning or coordination.
5. May be suitable for small systems or projects with a low degree of interdependence between components.

Disadvantages:

1. There will be quite a lot of delay because you would have to wait for all the modules to be integrated.
2. High-risk critical modules are not isolated and tested on priority since all modules are tested at once.
3. Not Good for long projects.
4. High risk of integration problems that are difficult to dentify and diagnose.
5. This can result in long and complex debugging

andtroubleshooting efforts.

6. This can lead to system downtime and increaseddevelopment costs.

7. May not provide enough visibility into the interactions and data exchange between components.
8. This can result in a lack of confidence in the system's stability and reliability.
9. This can lead to decreased efficiency and productivity.
10. This may result in a lack of confidence in the development team.
11. This can lead to system failure and decreased user satisfaction.

2. Bottom-Up Integration Testing – In bottom-up testing, each module at lower levels are tested with higher modules until all modules are tested.

3. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers todrive and pass appropriate data to the lower-level modules.

Advantages:

- In bottom-up testing, no stubs are required.
- A principal advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.
- It is easy to create the test conditions.
- Best for applications that uses bottom up design approach.
- It is Easy to observe the test results.

Disadvantages:

- Driver modules must be produced.
- In this testing, the complexity that occurs when the system is made up of a large number of small subsystems.
- As Far modules have been created, there is no working model can be represented.

4. Top-Down Integration Testing – Top-down integration testing technique is used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

Advantages:

- Separately debugged module.

- Few or no drivers needed.

- It is more stable and accurate at the aggregate level.
- Easier isolation of interface errors.
- In this, design defects can be found in the early stages.

Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.
- It is difficult to observe the test output.
- It is difficult to stub design.

5. Mixed Integration Testing – A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested.

6. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches. It is also called the hybrid integration testing.also, stubs and drivers are used in mixed integration testing. **Advantages:**

- Mixed approach is useful for very large projects having several sub projects.
- This Sandwich approach overcomes this shortcoming of the top-down and bottom-up approaches.
- Parallel test can be performed in top and bottom layer tests.

Disadvantages:

- For mixed integration testing, it requires very high cost because one part has a Top-down approach while another part has a bottom-up approach.
- This integration testing cannot be used for smaller systems with huge interdependence between different modules.

Applications:

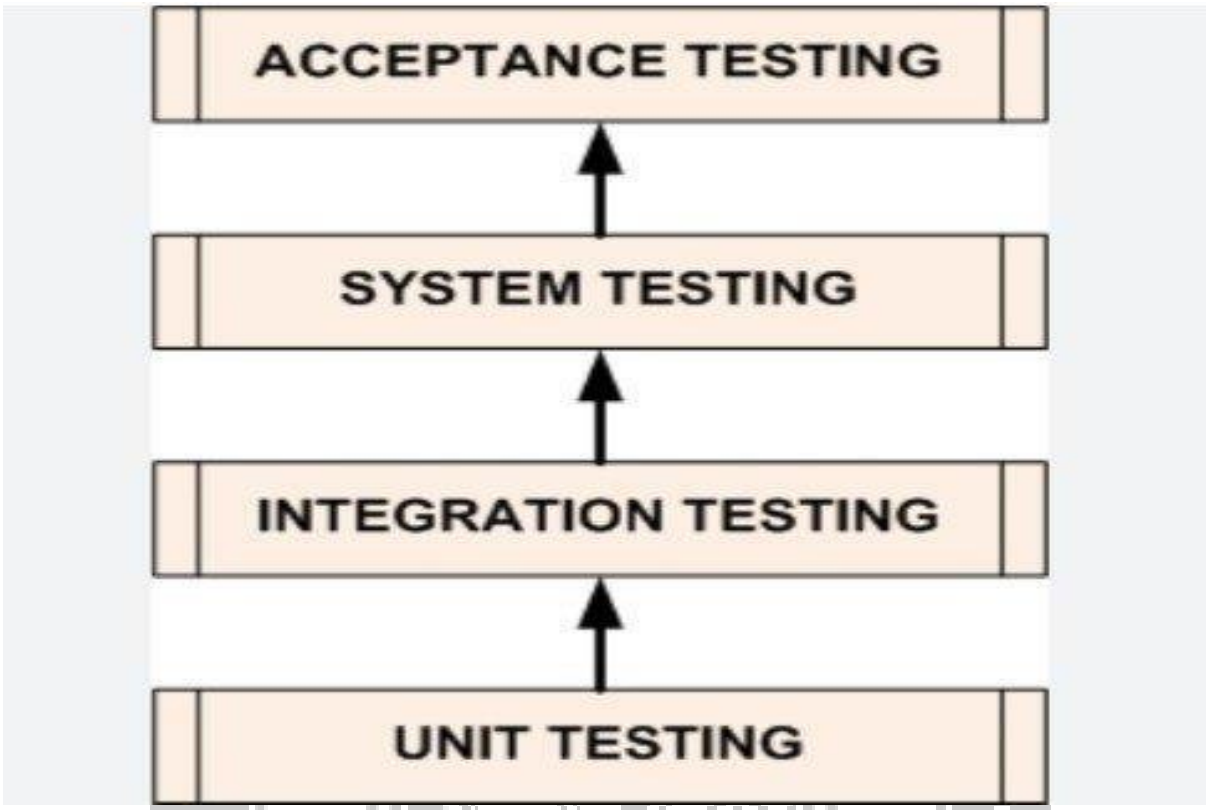
1. **Identify the components:** Identify the individual components of your application that need to be integrated. This could include the frontend, backend, database, and anythird-party services.
2. **Create a test plan:** Develop a test plan that outlines the scenarios and test cases that need to be executed to validate the integration points between the different components.

This could include testing data flow, communication protocols, and error handling.

3. **Set up test environment:** Set up a test environment that mirrors the production environment as closely as possible. This will help ensure that the results of your integration tests are accurate and reliable.
4. **Execute the tests:** Execute the tests outlined in your test plan, starting with the most critical and complex scenarios.Be sure to log any defects or issues that you encounter during testing.
5. **Analyze the results:** Analyze the results of your integration tests to identify any defects or issues that need to be addressed. This may involve working with developers to fixbugs or make changes to the application architecture.
6. **Repeat testing:** Once defects have been fixed, repeat the integration testing process to ensure that the changes havebeen successful and that the application still works as expected.

System testing

A type of Software testing, System testing comes at the third level after Unit testing and Integration testing. The goal of the system testing is to compare the functional and non-functional features of the system against the user requirements.



Stages of Software Testing

Once all the sub-systems or modules integrate to build one application, testers perform System testing to check for potential functional and non-functional irregularities. Overall, System testing checks for the system's design and behavior as per customer
Expectations. It is a black box testing technique.

Often, your QA team will rely on System Requirement Specifications (SRS), Functional Requirement Specifications (FRS),
or a mix of both. We will discuss these two options later in this blog.

Types of System Testing

System testing covers the end-to-end analysis of a software
application in terms of its functionality and user experience. Hence, you can categorize it into two parts—functional and non-functional.

Functional Testing

Functional testing validates the functional aspects of a software
system against the user requirements and functional specifications. Itchecks for user experience or interface, API integration, database,

security and privacy, and server communication, as mentioned in the user document.
One example of functional testing is checking for the login functionality using the right credentials. The system should not allow you to sign in if your username and password are wrong or not present in the database.

Non-Functional Testing

Non-functional testing checks for non-functional aspects of software, such as performance, reliability, usability, and application readiness. It intends to assess a system's performance per the non-functional conditions that never appear in the functional tests. Often, non- functional testing is important to check for security, application load capability, and utility to measure user satisfaction.
One example of non-functional testing is checking for the number of users the system can handle at a time. It helps to determine theperformance and usability of the system under high traffic.

Advantages of System Testing

- End-to-end test that involves all the software components as a whole and checks for defects
- Examine the software from a user point of view and simulate real-life scenarios
- Covers both functional and non-functional testing elements such as performance, usability, regression testing, and more
- No need for internal code knowledge

Differences between System Testing and Integration Testing

	System Testing	**Integration Testing**
Purpose	Testing the entire software as a whole and understanding if it meets the user requirements	Testing several modules together and understanding how they interface with each other
Executed by	Test Engineers	QA Testers, Developers and Test Engineers
Type of testing	Both functional and non-functional tests such as performance, security, regression testing, unit testing, etc	Only functional tests. Can be performed in different approaches like top-down, bottom-up, big-bang, and more
Technique	Black-box testing	Grey-box testing
When to perform	After Integration Testing	Before System Testing and after Unit Testing

		• Helps to figure out how different modules communicate with each other • Finds defects in the interface between various components
Advantages	Involves all the software components andchecks for bugs Examine the software from a user point of viewand simulate real-life scenarios Covers both functional and non-functional testing elements No need for internalcode knowledge	Can be performed bya wider variety of staff members. There are different approaches with different benefits toconsider when performing integration testing