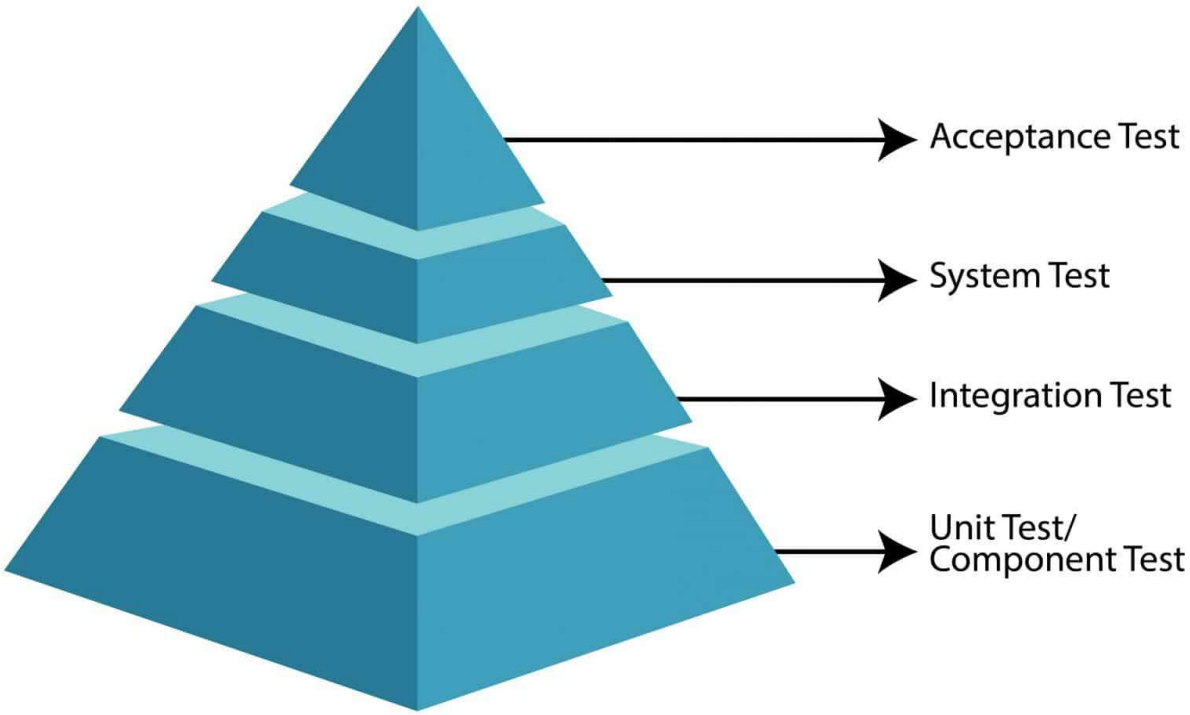
System Testing means testing the system in its entirety. In other words, all modules/components are integrated to verify if the system works as expected or not.

The performance of the system test happens after the ***Integration tests***. It plays an essential role in delivering a high-quality product. In this article, we are going to cover:-

What is System Testing?
Objectives of System Testing
What is the Test basis for System testing?
What are the Test Objects for System Testing?
Which are the typical Defects and Failures for System Testing?
Approach and Responsibilities
What are Different Types of System Testing?

What is System Testing?

System testing is a testing level that evaluates the behavior of a fully integrated software system based on predetermined specifications and requirements. It is a solution to the question "***if the complete system works according to its predefined requirements?***"



Therefore, some of the critical considerations for System testing are:

Firstly, the performance of System testing happens in a fully developed and integrated system.
Secondly, the performance of System tests happens on the entire system in the context of the functional requirements specifications (FRS) or the system requirements specifications (SRS), or both. In other words, System tests validate not only the design but also the behavior and usability aspects.
In addition to the above, it verifies the entire product, after integrating all software and hardware components and validating it according to the specifications.
Moreover, System testing can include both functional and non-functional types of testing.

We will try to understand the concept with the help of an example:

Let us take the case of a car manufacturer. A car manufacturer does not produce the car as a complete car. Each car component is manufactured separately, such as seats, steering, mirror, brake, cable, motor, car frame, wheels, etc. After the production of each item, independent testing happens to see if they are working the way they are supposed to work. It is called ***Unit testing****.*

After assembling each part, verification happens. It checks whether the assembly has not produced any side effects on the functionality of each component. Additionally, checking of smooth working of both the components also happens. It is called ***Integration testing****.*

Once all the parts are assembled, and the Car is in place - Can we safely assume that the car is ready to drive? The entire Car must be checked for the different aspects according to the defined requirements, as if:

The car can be operated smoothly, the brakes, gears, and other functions work correctly (Functional Testing).
The Airbags will come out in case of a crash (Non-Functional Testing).

And all this test effort is called System Testing, which verifies the car on every aspect.

Once the car is assembled, and ready for use, do we just roll it out to the public? No, we have another test level called ***User Acceptance testing****, where a group of Users/Customers will test the car in real-life conditions. They will drive the car on the road, see how the car performs in terms of overall comfort, experience, and key features like Brakes, Gears, music system, etc. Once UAT stage is passed, then the Car is ready to be rolled out to the customers. We will learn more about UAT in our subsequent articles.*

Objectives of System Testing

The primary objectives of System testing are as below:

One of the primary objectives of System testing is to reduce risk. Even after individual testing of components, risk of how they will all come together to form a complete System still exists. System testing eliminates this risk by ensuring that it will function as per customer requirements.

System testing must verify whether the design of the functional and non-functional behaviors of the system is as per the customer's specifications.
Validate that the system is complete and will work as expected.
System testing aims to build confidence in the quality of the system as a whole.
System testing also aims to find defects and to prevent defects from escaping to higher test levels or production. Additionally, it is the only phase that occurs on the full System just before the User Acceptance testing. So it's critical to find all the possible defects at this stage, and they don't leak to production.
System Testing results are used by stakeholders to make release decisions. The Entry criteria for User Acceptance testing is the basis completion of System Testing. System testing may also adhere to legal or regulatory requirements or standards.

What is the Test basis for System testing?

The test basis is the source of information or the document, which is the main requirement for writing test cases and also for test analysis. The base of the test must be well defined and adequately structured so that one can quickly identify the test conditions from which the test cases are derived.

Examples of work products used as a test basis for system tests include:

System and software requirements specifications (*functional and non-functional*) - *SRS gives complete requirements on how the integrated System should work. It should form the basis of coming up with System Test Scenarios*.
Risk analysis reports - *It indicates areas that are risky. It could be from the implementation perspective or legal/compliance perspective. System tests should ensure that the focus is on these areas*.
Use cases - *They show the journey flows of the System. It forms the basis of created end to end scenarios*
State diagrams - *This is visual representation in the form of flow charts of how each component interacts with each other and their trigger points*.
Models of system behavior - *This describes the processes and activities that each component is involved in, and also shows how they will interact with other components*.
System and user manuals - *Often for a product based software, user manuals are created, so it's easy for the user to figure out the usage. E.g., for an income tax calculation software, user manual will describe how to fill the data, and how the calculation takes place*
Epics and user stories - *Epics and user stories will give a high-level view of the System. A combination of them creates an end to end system test cases*.

What are the Test Objects for System Testing?

Test Objects are the component or systems that require testing. Now let's look at the test objects for system testing:

Typical Test Objects of System test include:

Applications

Hardware/Software Systems
Operating Systems
System under Test
System configuration and configuration data

If you look at these Test Objects, you will notice that these are fully integrated Systems. The System could be a software (*like Amazon or Flipkart website/app*), or it could be an Operating System (*like Windows 10*), etc.

Which are the typical Defects and Failures for System Testing?

The System test is usually the final test from the software development team. It ensures that the system delivered finally will meet the specification. In addition to that, its purpose is to find as many defects as possible before it goes to the next level of ***User Acceptance testing***.

Examples of defects and typical failures for System tests include:

Failure to carry out end to end functional tasks - E.g., A train ticket booking software successfully books the ticket but fails to send the customer a confirmation email with their ticket number.
Incorrect calculations - E.g., Consider that you are shopping on the Amazon website. You have added two products worth $100 and $50. The cart value shows as $150. Now you apply a 10% discount on the Cart which gives you a discount of $15 and brings the Cart value down to $135. After placing the order, you decide to cancel the $50 product. The cancellation of the product happens, and you get a refund of $50. Its a System failure that applied a10% discount whereas the actual refund should have been $45.
Incorrect control or data flows within the system. E.g., Consider that you have used a discount of 15% on a product which is worth $5. After the discount, the value comes out to be $4.75. Now the application has a wrong logic of rounding off at first decimal place, where it charges the card $4.8 instead of $4.75. Such incorrect data flow could lead to defects in System testing phase
Unexpected functional or non-functional behavior of the system - E.g., you are using the Amazon app, and if there is an incoming call, the application is crashing. While there is nothing functionally wrong with the app, fixing such non-functional behaviors is critical to the overall success of the application.
There are situations where the system doesn't work as described in the system and user manuals.
There are situations where the system fails to work correctly in production environments. Often the System works perfectly fine in test environments, but upon it's release to a production-like environment, it fails to work. Therefore, a System test must always happen in an environment that mimics production; in terms of software and hardware, simultaneously.

Approaches and Responsibilities

Independent testers often do System testing to have an objective view of the System. The strategy is to involve the testing team from the start, so they have a complete understanding of the System. System test scenarios are usually end-to-end tests that can cover the full System. Architects and product owners usually review these scenarios.

Responsibilities of System Testing Team:

Understand System flows, and create high-level user journeys
Create Detailed End to End System test cases
Identify and generate any test data required to execute the test cases
Coordinate with Scrum teams to ensure adequate support is available to fix defects
Defect triage meetings to fix accountability of a defect fix to a Scrum team

What are Different Types of System Testing?

Like any software test, System tests are also an amalgam of various test types, which allow the team to validate the overall performance and functionality of the product. Each of these test types focuses on different aspects of the product and satisfies the various requirements of the client/user. These types of system tests are:

***Usability testing**: Usability tests mainly focus on the user's ease of using the application, the flexibility in handling controls and the ability of the system to meet its objectives*
***Load testing**: The load test is necessary to know that a software solution will work under real-life loads*.
***Regression testing**: Regression tests involve tests performed to ensure that none of the changes made during the development process have caused new errors. It also ensures that no old errors appear when adding new software modules over time*.
***Recovery testing**: Recovery testing happens to demonstrate that a software solution is reliable and can successfully recover from possible crashes*.
***Migration testing**: Migration testing happens to ensure that the software moves from older system infrastructure to current system infrastructure without any problem*.
***Performance testing**: We do Performance Testing to examine the response, stability, scalability, reliability, and other software quality metrics, under different workloads*.
***Security testing**: Security Testing evaluates the security features of the software to guarantee, protection, authenticity, confidentiality, and integrity of information and data*.

The performance of System tests happens once the software development process completes. In addition to that, it happens after the product has gone through the Unit and Integration tests. It is an integral part of the software testing life cycle. This test is not limited to one aspect or component of the product. But it tests the software system as a whole which makes it an essential part of any successful test cycle.