1. **what is software testing?**

**Answer :** Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

1. **What is Exploratory Testing?**

**Answer :** During testing phase where there is severe time pressure, Exploratory testing technique is adopted that combines the experience of testers along with a structured approach to testing.

Exploratory testing often performed as a black box testing technique, the tester learns things that together with experience and creativity generate new good tests to run.

Benefits:

Following are the benefits of Exploratory Testing:

Exploratory testing takes less preparation.

Critical defects are found very quickly.

The testers can use reasoning based approach on the results of previous results to guide their future testing on the fly.

**Drawbacks:**

Following are the Drawbacks of Exploratory Testing:

Tests cannot be reviewed.

It is difficult to keep track of what tests have been tested.

It is unlikely to be performed in exactly the same manner and to repeat specific details of the earlier tests.

1. **What is traceability matrix?**

**Answer** : A Traceability Matrix is a**table that helps you trace project related artifacts, both forward and backward**. Project related artifacts include requirements, designs, test cases, test steps, defects, and so on. Using a Traceability Matrix, you can trace one of these artifacts to all other linked artifacts.

1. **What is Boundary value testing?**

**Answer :**

**This one of the software testing** technique in which the test cases are designed to include values at the boundary.

If the input data is used within the boundary value limits, then it is said to be positive testing

If input data is picked outside the boundary value limits, then it is said to be negative testing .

* Example : A system can accept the numbers from 0 to 10 numeric values.
* All other numbers are invalid values. Under this techniques, boundary values 0,10 and and -10 will be tested .

1. **What is Equivalence partitioning testing ?**

**Answer :**

* Aim is to treat groups of inputs are equivalent and to select one representative input to test them all .
* EP can be used for all levels of testing .
* Equivalence partitioning is the process of defining the optimum number of tests by :
* Reviewing documents such as the functional design specification and detailed design specification, and identifying each input condition within a function.
* Selecting input data that is representative of all other data that would likely invoke the same process for the particular condition.
* The numbers fall in to a partition where each would have the same or equivalent, result i.e, an equivalence partition (EP) or equivalence class.
* Ep says that by testing just one value we have tested the partition (typically a mid-point value is used). It assumes that.
* If one value finds bug , the others probably will too
* If one doesn’t find a bug, the others probably won’t either.
* In EP we must identify valid equivalence partitions and invalid equivalence partitions where applicable (typically in range tests)
* The valid partition is bounded by the values 1 and 100.
* Plus there are 2 invalid partitions.

1. **What is Integration testing?**

Answer : Integration testing is a type of testing meant to check the combinations of different units, their interactions, the way subsystems unite into one common system, and code compliance with the requirements.

* For example, when we check login and sign up features in an e-commerce app, we view them as separate units. If we check the ability to log in or sign up after a user adds items to their basket and wants to proceed to the checkout, we check the integration between these two functionalities.
* For integration testing, the team uses components that have already been tested as separate units. A QA team groups these units into sets and checks them in accordance with the test plan.
* Integration testing is performed using the black box method. This method implies that a testing team interacts with an app and its units via the user interface – by clicking on buttons and links, scrolling, swiping, etc. They don’t need to know how works or consider the backend part of the code components**.**
* Integration testing and unit testing are two [levels of software testing](https://u-tor.com/topic/software-testing-levels), where a unit is the basic one and integration is the sequential one.
* Unit testing implies checking the smallest functioning parts of code separately. Integration testing goes further: we look for errors that happen when units start interacting. The fact that units have successfully passed the tests at the previous stages doesn’t guarantee that they will work well together.
* Therefore, any problems arising when we assemble several smaller parts into a subsystem can be associated with the particularities of the interaction between these units. The earlier we notice something abnormal, the lower will be the cost of a mistake.
* Unit testing is the initial stage of the QA process. It prepares the functionality to the following stage which is integration testing. Without passing the former and verifying that units perform correctly, we cannot proceed to the latter and start putting them together.
* Related: [Importance of Unit Testing: How It Aids Development Projects](https://u-tor.com/topic/unit-testing-importance)
* If integration testing is the first chapter of the story (or the pilot of a TV show), unit testing is a prequel. As a rule, software engineers run unit testing at the earlier stages of product development.
* To wrap it up, unit testing is the examination of the smallest functional parts of code. Integration testing is the examination of the smallest possible combinations of those parts.

1. **What determines the level of risk?**

Answer : A properly designed test that passes, reduces the overall level of Risk in a system

• Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’

• When testing does find defects, the Quality of the software system increases when those defects are fixed

• The Quality of systems can be improved through Lessons learned from previous projects

• Analysis of root causes of defects found in other projects can lead to Process Improvement

• Process Improvement can prevent those defects reoccurring

• Which in turn, can improve the Quality of future systems

• Testing should be integrated as one of the Quality assurance activities

A Risk could be any future event with a negative consequence .You need to identify the risks associated with your project

• Risks are of two types

• Project Risks

• Product Risk

Example of Project risk is Senior Team Member leaving the project abruptly.

• Every risk is assigned a likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10 .

• But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place

• Example of product risks would be Flight Reservation system not installing in test environment

• Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing.

1. **What is Alpha testing?**

**Answer:** An alpha test is a form of acceptance testing, performed using both black box and white box testing techniques. As it is the first round of testing a new product or software solution goes through, alpha testing is concerned with finding any possible issues, bugs or mistakes, before progressing to user testing or market launch.

The main purpose of conducting an alpha test is to ensure the quality of the software system before it goes into the production environment. That’s why an alpha test relies on internal testers — team members, [stakeholders](https://airfocus.com/glossary/what-is-a-stakeholder/), etc. — at the developer's site, in a virtual environment similar to the actual production environment.

You may have also heard of beta testing, which takes place after alpha testing and is performed by potential end-users.

1. **What is beta testing?**

**Answer :** Beta testing is a type of user acceptance testing where the product team gives a nearly finished product to a group of target users to evaluate product performance in the real world.

There is no standard for what a beta test should look like and how to set up beta testing.

The actual testing procedure should be relevant to your testing goals. However, there are a few requirements that a product needs to comply with in order to be ready for beta testing:

* The product should be in the “Feature complete” state (it should have all the features that are planned for the release version.
* The product should be stable (test participants should not face unpredictable crashes.
* Test participants should belong to the product’s target audience.
* Test participants should complete real-world scenarios while interacting with an app, and they should do it in the real environment (not in lab environments.

1. **What is component testing?**

**Answer :** Component testing is a type of white box testing where you validate an individual component of the application before testing the entire application. As a consequence, component testing finds bugs and verifies the functionality of software modules/programs which are individually testable.

1. **What is functional system testing?**

**Answer :** **FUNCTIONAL TESTING** is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

## What do you test in Functional Testing?

The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on –

* **Mainline functions**:  Testing the main functions of an application
* **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
* **Accessibility**:  Checks the accessibility of the system for the user
* **Error Conditions**: Usage of testing techniques to check for error conditions.  It checks whether suitable error messages are displayed.

**How to do Functional Testing**

Following is a step by step process on **How to do Functional Testing** :

* Understand the Functional Requirements
* Identify test input or test data based on requirements
* Compute the expected outcomes with selected test input values
* Execute test cases
* Compare actual and computed expected results

1. **What is Non-Functional Testing?**

**Answer :** Non-functional Testing is a method of software testing that examines a software application's non-functional attributes (productivity, compatibility, functionality, and many more). It is intended to assess a system's preparation based on nonfunctional conditions that never get covered by functional tests.

A good example of a non-functional test is seeing how many users would sign into a program at the same time.

Non-functional testing is almost as critical as functional testing, and it has an effect on customer satisfaction.

We will discover the following in this tutorial−

* Non-functional testing objectives
* Non-functional testing Characteristics
* Testing for non-functionality variables
* Software Testing Types
* Types of Non-functional Testing
* Test Case Examples of Non-Functional Testing

Non-functional Testing Objectives

* Non-functional testing can improve the product's reliability, performance, manageability, and accessibility.
* Reduces the probability and cost of manufacturing related to noncomponents of the commodity.
* Improves the way the product is installed, configured, executed, controlled, and tracked.
* Measurements and metrics should be collected and generated for internal analysis and development.
* Develops and reinforces your understanding of the product's actions and the technology that is being used.

Non-functional testing Characteristics

* Non-functional testing ought to be quantitative, but arbitrary characterizations such as fine, great, greatest, and so on have no room.
* At the outset of the specification procedure, exact quantities are impossible to be determined.
* It is essential to keep the specifications as the topmost priority.
* Guarantee the consistency characteristics are properly defined in Software Engineering.

Testing for non-functionality variables

* **Security:** The parameter specifies how a system is protected from intentional and unintentional threats by both internal and external sources. Security Testing is used to validate this.
* **Reliability:** The degree to which a software system executes the required tasks consistently and without error. Reliability Testing is used to validate this.
* **Survivability:** The parameter ensures that the computing system runs smoothly and restores in the event of a system malfunction. Recovery Testing verifies this.
* **Availability:** The parameter specifies how much the user can rely on the machine when it is running. Stability Testing verifies this.
* **Usability:** The convenience with which a user can read, perform, and plan inputs and outputs by contact with a machine. Usability testing verifies this.
* **Scalability:** The concept refers to the extent to which a software application's computing power may be expanded to satisfy increased demand. Scalability Testing is used to validate this.
* **Interoperability:** This non-functional parameter validates the interactions of a software system with other software systems. Interoperability Testing verifies this.
* **Efficiency:** The degree to which a software system can manage power, volume, reply variations, and duration.
* **Flexibility:** The concept refers to how quickly and easily the program can run in various hardware and software setups.
* **Portability:** The ability of software to shift from its existing hardware or software setting.
* **Reusability:** It refers to a piece of software that can be adapted to be used in another program.

Software Testing Types

In general, there are three types of testing−

* Functional
* Non-Functional
* Maintenance

There are several testing levels in all categories of testing, but they are commonly referred to as Testing Types. The above definition can vary slightly in various books and study guides.

The following list is not exhaustive and there are over 100 different types of testing and increasing. No need to be concerned; you will learn them when you advance in the research industry. Also, keep in mind that not all research types are applicable to all programs, but rather rely on the purpose and scale of the project. More about this in a subsequent tutorial.

Types of Non-functional Testing

Following are the most common types of Non-Functional Testing −

* Performance Testing
* Load Testing
* Failover Testing
* Compatibility Testing
* Usability Testing
* Stress Testing
* Maintainability Testing
* Scalability Testing
* Volume Testing
* Security Testing
* Disaster Recovery Testing
* Compliance Testing
* Portability Testing
* Efficiency Testing
* Reliability Testing
* Baseline Testing
* Endurance Testing
* Documentation Testing
* Recovery Testing
* Internationalization Testing
* Localization Testing

1. **What is GUI Testing?**

**Answer :** Graphical user interface testing (GUI testing) refers to the testing of a graphical user interface to ensure trouble-free use and implementation.

A graphical user interface is a software tool that helps end users to utilize a software environment. These tools need to work well in order to support a user base.

Graphical user interface testing is built on specific paradigms that help developers to test the full range and functionality of the system. These are sometimes called "test cases." Developers and others need to check to make sure that each part of a graphical user interface is functioning correctly—for example, that use of each individual menu item, window, text box or other control works the way it is supposed to. With GUI testing, professionals look for a wide range of problems, from improper output and small bugs or glitches to complete system crashes.

Regression testing for GUIs involves multiple or complex control paths. For instance, a developer might have to check a specific series of user activities where a user might first select the menu item and then use other controls in sequence. The idea of complex control paths has numerous different permutations that need to be checked in GUI testing.

GUI testing also has to be done for each individual device environment. One major application of GUI testing has occurred as portable devices like smartphones and tablets have gradually taken over a lot of the functionality previously facilitated by laptop computers. The GUIs for mobile devices are different from those of laptop or desktop computers, and many different kinds of software teams needed to try to migrate GUI systems to phones accordingly.

The graphical user interface represents, in many ways, the core of the environment, and GUI testing generally takes significant work and investment on the part of tech providers.

1. What is Adhoc testing?

Answer : **Ad hoc Testing** is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

Following video guides you how to do adhoc testing

When execute Adhoc Testing?

Ad hoc testing can be performed when there is limited time to do elaborative testing. Usually adhoc testing is performed after the formal test execution. And if time permits, ad hoc testing can be done on the system. Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

1. **What is white box testing and list the types of white box testing?**

Answer : **White Box Testing** is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “[Black Box Testing](https://www.guru99.com/black-box-testing.html)” symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

**White box testing involves the testing of the software code for the following**:

* Internal security holesx
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object, and function on an individual basis.

1. **What is black box testing? What are the different black box testing techniques?**

Answer : Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

An example of a [security technology that performs black box testing is Dynamic](https://www.imperva.com/learn/data-security/information-security-infosec/)[Application Security Testing](https://www.imperva.com/learn/application-security/application-security/)(DAST), which tests products in staging or production and provides feedback on compliance and security issues.



Black box testing can be applied to three main types of tests: functional, non-functional, and regression testing.

Functional Testing

Black box testing can test specific functions or features of the software under test. For example, checking that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials.

1. Mention what are the categories of defects?

Answer:

* Wrong: If the requirements are implemented incorrectly, then they are stated as Wrong defects.
* Missing: If the requirement is not done which is given by the customer. ...
* Extra: If a requirement is not given by the end user and if it is done,then it is called as an extra defect.

1. Mention what bigbang testing is?

# Answer :

## Definition of Big Bang Testing:

Big Bang Testing is an approach of integration testing where integration of all or major components of the system are tested. The Big Bang method is very effective for saving time in the integration testing process. It helps amateur developers to find integration related defects earlier than actual integration testing

1. **What is the purpose of exit criteria?**

Answer : Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

**Examples of Exit Criteria**

Verify if All tests planned have been run.

Verify if the level of requirement coverage has been met.

Verify if there are NO Critical or high severity defects that are left outstanding.

Verify if all high risk areas are completely tested.

Verify if software development activities are completed within the projected cost.

Verify if software development activities are completed within the projected timelines.

1. **When should "Regression Testing" be performed?**

**Answer :** Software development is a highly complex process of creating and maintaining a product that would solve its users’ problems with ease and finesse. In addition to market research, requirement analysis, and other business-related procedures, [**software development**](https://qarea.com/services/custom-software-development)mainly involves a lot of writing the source code and maintaining it in good health. Software development is impossible without code changes, making code quality a real struggle—especially when you’re faced with a tough deadline—as every minor change can have an unexpected, even unseen impact on the rest of your codebase.

In quality assurance, these bugs are called **software regressions**, defects that appear in the course of development and can cause your entire software solution or its separate features to stop functioning as intended.

There are three types of regressions:

* Local regressions take place when a change to a piece of code introduces a new bug and causes a given component to stop working correctly.
* Remote regressions happen when a change in one part of the code causes an error, or completely breaks functionality, in another part of the software that had no issues before the change.
* Unmasked regressions occur when a change to a piece of code reveals an already existing bug that previously did not affect the software in any way. This type of regression is the most dangerous: a ticking bomb, unexpected, and thus the hardest to find.

Whether your software is early in development or you’re introducing an update to a released solution, regression testing comes in to ensure that any changes you make to your code do not cause you any pitfalls.

1. **What is 7 key principles? Explain in detail ?**

**Answer** : If you have heard about data privacy then you have no doubt heard about the GDPR. There are 7 key principles that are the foundation of the GDPR, so what are they?

* Lawfulness, fairness and transparency
* Purpose limitation
* Data minimisation
* Accuracy
* Storage limitation
* Integrity and confidentiality (security)
* Accountability
* These principles are set out at the very beginning of the legislation and are the building blocks for the rest of it. They are what your Privacy Policy needs to be based on in order to ensure it is GDPR compliant. Let's take a look in a little more depth at each of these key principles.

1. **Difference between QA v/s QC v/s Tester ?**

**Answer :** Quality assurance is a term used to signify a practice aimed at preventing mistakes in products and services, but the term or the abbreviation often refer to the person in charge of it, so instead of quality assurance manager (QAM), you may hear people talking of quality assurance when describing a job profile. This position is often considered to be synonymous with the position of a tester, but there are significant differences.

Saying that tester and QA are the same, is just like saying that a developer and an analyst do the same job, or that a paralegal and judge are two words to describe the same position, just because both are dealing with legal matters.

Testing is an activity. Anybody can test. Sometimes testing is just using a product, while QA requires a strategic approach to testing in the sense of planning how, what, and when to test.

Imagine that you're good at cooking and that you can prepare tasty meals. Given that you cook for your family and friends, does that make you capable to run a restaurant? Could you plan and design the menus, source the supplies, monitor the food and the service, train your staff, build brand awareness, know which dishes would attract new diners and so on? The ability, or knowledge how to cook and the passion is just a small, albeit an integral part of a much more complex and dynamic process that is running a restaurant. To be a successful restaurant owner and manager you need to be able to plan things strategically and execute them tactically, and perhaps most importantly, know how to handle the pressure. Same applies to quality assurance, testing is just one, although perhaps the most important part of it.

Let's call things by their name

Somewhere I read the following phrase and it really caught my attention:

"A Tester is responsible for finding failures, but a QA not only finds them but helps to prevent them."

Without going deeply into the tasks performed by each of them, this may be the best way to describe the distinction between a tester and a QA. After reading various definitions of the two positions, I find this to be the simplest, yet most accurate way to describe them:

**Tester :**a person in charge of testing software during its development phase in order to detect bugs and report them.

**Quality Assurance Specialist:**a person who performs a set of activities in order to ensure the quality of software during all phases.

The key difference is that the QA is interested in the process while the Tester is interested in the product. The difference may seem subtle, but it is rather significant. A QA is a lot more goal-oriented and is always concerned with the bigger picture.

1. **Difference between Smoke and Sanity?**

# **Answwr : Difference Between Smoke and Sanity Testing**



The smoke and sanity testing are techniques work as a part of the integration and regression testing respectively. The significant difference between the smoke and sanity testing is that the smoke test is employed in the unstable product while the sanity testing is applied to the more stable products. The smoke testing can be said to be shallow testing as it just tests for the vital requisite, but sanity testing examines each of the modules of the software at the end, for checking whether the applied changes works well.

| BASIS FOR COMPARISON | SMOKE TESTING | SANITY TESTING |
| --- | --- | --- |
| Basic | Smoke testing evaluates and tests for the essential functions. | Sanity testing checks the software modules deeply. |
| Test cases | Can be written or automated test. | Unscripted |
| Approach | Shallow and wide | Narrow and deep |
| Prominence | Chief purpose is to cover every part of the application swiftly. | Emphasis on the functioning of the modules (software parts) of the software. |
| Acted upon | Every build | Only on the stable build. |
| Performed by | Developer | Tester |

**Definition of Smoke Testing**

The **Smoke testing** is mainly originated from the integration testing approach. It usually starts before full-scale testing which covers a broad part of the software but not more complex and detailed aspects of it. Smoke testing is considered as the non-exhaustive testing where the working of the most crucial features of the product is checked.

**The activities performed in the smoke testing are:**

* Firstly, it converts the software modules into the code and collaborates it into a “build”. A build is consist of the data files, reusable modules, libraries, and engineered components which are needed to employ one or more functions.
* A series of test cases are planned to discover errors for making sure the functions perform appropriately.
* The multiple builds are then integrated into a single product, and the whole product is smoke tested repeatedly.
* The testing process is continued only until the results comply with the fundamental requirement of the product, but if the results do not match to the basic requirements, the product is returned to the development team for the essential changes.
* Advantages of Smoke Testing
* Risk minimisation by detecting and rectifying the defects earlier.
* Repeated inspection improves the quality of the system.
* Simplified error determination and correction
* Progress is easily evaluated

**Definition of Sanity Testing**

The **Sanity testing** is a way to test the build entirely after applying the small alterations in the code and the functionality. It primarily checks whether the product is working correctly after alterations and the bugs have been fixed or not. The sanity testing is the subgroup of the regression testing and performed before the launching of the product. If the proposed functionality does not work according to expectations, the composed build is discarded to eliminate the time and cost needed in the rigorous testing.

The software needs to go through the other testing before undergoing to the sanity testing. This type of test is deeper in sense, means that it considers the detailed aspects of the software.

**Advantages of Sanity Testing**

* Good time utilization as the focus is on one or few areas of the functionality.
* Assures the proper working of the application after the implication of the slight changes in the code.
* Discovers the dependent missing objects.

1. **Difference between verification and Validation ?**

**Answer : Verification in Software Testing** is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.

**Validation in Software Engineering** is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not. The process helps to ensure that the software fulfills the desired use in an appropriate environment. The validation process involves activities like unit Here is the main difference between verification and validation testing:

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| The verifying process includes checking documents, design, code, and program | It is a dynamic mechanism of testing and validating the actual product |
| It does **not** involve executing the code | It always involves executing the code |
| Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc. | It uses methods like Black Box Testing, [White Box Testing](https://www.guru99.com/white-box-testing.html), and non-functional testing |
| Whether the software conforms to specification is checked | It checks whether the software meets the requirements and expectations of a customer |
| It finds bugs early in the development cycle | It can find bugs that the verification process can not catch |
| Target is application and software architecture, specification, complete design, high level, and database design etc. | Target is an actual product |
| QA team does verification and make sure that the software is as per the requirement in the SRS document. | With the involvement of testing team validation is executed on software code. |

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

**Answer :** Testing is the process of identifying defects, where a defect is any variance between actual and expected results. “**A** **mistake** **in** **coding** is called **Error**, **error** found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is **Failure**.” **DEFECT**:

1. **Difference between Priority and Severity ?**

**Answer** : Severity is a measure of the potential business impact of an issue. Severity is usually classified into four levels: Critical, Major, Minor, and Trivial. Critical issues have the potential to cause serious business disruption, while major issues may cause significant inconvenience or some business disruption. Minor issues are generally inconveniences, while trivial ones have little or no impact. When troubleshooting an issue, it is important to accurately assess the Severity in order to determine the appropriate course of action. If an issue is Severity 1 (Critical), then it requires an immediate fix in order to avoid significant business disruptions. For Severity 2 (Major) issues, a workaround may be possible in order to minimize the impact on business operations. For Severity 3 (Minor) and 4 (Trivial) issues, the goal is usually to resolve the issue as soon as possible, but there is usually less urgency than for Critical or Major issues.

## Priority

## Priority determines how fast or how perfectly the bug is eradicated. Priority is set according to the needs of the project.

* For example, a showstopper bug found in the production environment would have the highest priority whereas a low-impact cosmetic bug found in the development environment would have the lowest priority. All else being equal, a bug that can be fixed with a one-line code change is a lower priority than a bug that requires significant code refactoring.
* Priority can also be set by considering how difficult it would be to work around the bug. A very simple workaround might make a bug of high priority low priority. Priority also takes into account whether the fix is likely to cause side effects.
* A fix for one part of the system that causes breakage in another part of the system would generally be a lower priority than a fix that does not have any side effects. Finally, business considerations such as expected return on investment (ROI) can also play a role in setting priority.

## Severity and priority are two important factors that need to be considered when troubleshooting an issue. Severity is a measure of the impact of the issue, while priority is a measure of the urgency of the issue. For example, a Severity 1 issue is one that has a major impact on production, while a Severity 2 issue is one that has a minor impact on production. A Priority 1 issue is one that needs to be fixed immediately, while a Priority 2 issue is one that can be fixed at a later time. When troubleshooting an issue, it is important to consider both severity and priority in order to determine the best course of action.

1. **What is Bug Life Cycle?**

**Answer : Even the most carefully planned and developed software is bound to have issues. While your development team can do its best to prevent these issues from occurring in the first place, perfection is almost impossible. What’s more important is knowing how to deal with bugs when they inevitably appear. This requires understanding the bug life cycle.**

**The bug life cycle, which is also called the defect life cycle, refers to the various states of a software issue, from initial identification to resolution. Developers classify bugs into life cycle stages so they can clearly and systematically communicate about the software bugs they deal with when developing and updating a software program.**

There are typically two primary participants in the defect life cycle: the developer and the tester. The tester either identifies a bug or, in the case of a user identifying a bug, reviews the bug before assigning it to the development team. A developer then triages the issue and figures out a solution.

In some cases, especially with small development teams, there may not be dedicated testers. Instead, one or more developers will perform both roles. While this may be unavoidable for some teams due to budget or other constraints, many experts in this space recommend having both roles on the team for several reasons.

First, the two roles have different perspectives. Developers focus on creating solutions while testers focus on breaking them. In addition, having developers test their own work is similar to someone editing their own writing — it’s harder to see your own mistakes.

Regardless of your team’s composition, the bugs you’ll deal with will go through the defect life cycle below.

## Breakdown of the bug life cycle

### **1. New**

This is the initial state of any bug that’s been identified, whether by a tester, a user, or a developer. The tester performs an initial review of the bug and creates a bug report to keep track of the issue.

### **2. Assigned**

The appropriate lead person — lead tester, project manager, or similar — assigns the bug to a developer, who takes responsibility for fixing it.

### **3. Open**

This is where the developer analyzes the bug and attempts to find a fix. Depending on the complexity of the bug, it may stay in this state for some time.

While the developer’s ultimate goal is to fix the defect, they may need to classify the bug as one of several substates, depending on their analysis:

* **Rejected.** In this case, the developer has determined the system is working according to specifications and the bug isn’t a genuine defect. For example, whoever reported the bug may have misinterpreted the result.
* **Duplicate.** The developer has identified the bug as a copy or reasonable facsimile of another bug. Both will be resolved when one is fixed.
* **Deferred.** Developers may defer bugs for a number of reasons, including lack of time to address the issue or a seemingly insignificant impact on functionality.
* **Not reproducible.** If the developer cannot reproduce the bug, they may send it back to the tester to get more accurate reproduction steps. The tester can then reassign the bug to the developer.

### **4. Fixed**

Once the developer produces and verifies a fix, they mark the issue as resolved and send it back to the tester.

### **5. Pending retest**

The bug idles in this state while waiting to be tested again.

### **6. Retest**

Once the tester begins trying to reproduce the bug, it goes into this state.

### **7. Reopen**

If the tester finds that the bug persists despite the developer’s fix, the tester reassigns it to the developer for reassessment.

### **8. Verified**

Assuming the tester cannot reproduce the bug and finds that the developer has truly fixed the issue, the tester marks the status as verified.

### **9. Closed**

After verifying the bug as fixed, the tester closes out the issue.

1. **Explain the difference between Functional testing and NonFunctional testing ?**

#### **Answer : key Difference Between Functional Testing and Non Functional Testing**

* Functional testing verifies each function/feature of the software whereas Non Functional testing verifies non-functional aspects like performance, usability, reliability, etc.
* Functional testing can be done manually whereas Non Functional testing is hard to perform manually.
* Functional testing is based on customer’s requirements whereas Non Functional testing is based on customer’s expectations.
* Functional testing has a goal to validate software actions whereas Non Functional testing has a goal to validate the performance of the software.
* A Functional Testing example is to check the login functionality whereas a Non Functional testing example is to check the dashboard should load in 2 seconds.
* Functional describes what the product does whereas Non Functional describes how the product works.
* Functional testing is performed before the non-functional testing.

## What is Functional Testing?

Functional testing is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing, and it is not concerned about the source code of the application.

Every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test. The testing can be done either manually or using automation

## What is Non-Functional Testing?

Non-functional testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application. It is explicitly designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

A good example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Functional** | **Non-functional testing** |
| **Execution** | It is performed before non-functional testing. | It is performed after the functional testing. |
| **Focus area** | It is based on customer’s requirements. | It focusses on customer’s expectation. |
| **Requirement** | It is easy to define functional requirements | It is difficult to define the requirements for non-functional testing. |
| **Usage** | Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| **Objective** | Carried out to validate software actions. | It is done to validate the performance of the software. |
| **Requirements** | Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| **Manual testing** | It describes what the product does Functional testing is easy to execute | It’s very hard to perform non-functional testing manually. |
| **Functionality** | Ch execute by manual testing eck login functionality. | It describes how the product works. |
| **Example Test Case** | Check login functionality. | The dashboard should load in 2 seconds. |

1. **To create HLR & TestCase ?**

**Answer :** Software terminology can sometimes be confusing. Test case, test script, test scenario, test plan: they all sound alike, but all have different functions and meanings for the entire software development cycle (SDLC). In the SDLC, creating and running test cases is the first of several steps within the testing process, laying the foundation for releasing a high-quality product.

Test cases are used to check whether the product meets specified requirements and established quality parameters (e.g. functionality, usability) prior to release.

Writing test cases, especially for QA/test managers, often causes headaches because it requires a lot of expertise to describe precise, typical scenarios and test conditions with realistic, goals. During the creation of test cases, you have probably already encountered the following questions:

* What is my **starting point** for the test case? Which prerequisites must be met?
* Will the test case reveal **the required results**? What are the primary areas?
* What **data** is needed for a successful test case?
* Am I able to **formulate the steps** and expected results clearly?
* Is the **structure** of the test case well planned? Does the tester / user know what to expect?
* Will the test cases **save time**, optimize existing processes and thus generate sustainable added value?

You can create great test cases by following these four steps.

### 1. Define the test scope

Make sure you’re as detailed as possible without overloading or confusing the tester. Whether you’re writing for an external crowd or internal audiences, you need to make sure the tester is “examining” the test object as if they are seeing it for the first time and has no background knowledge. The real art of development is to make the test environment so intuitive that it seems familiar to the tester even when they’re first encountering it.

When preparing the scope and its test cases, pay attention to the following points:

* Explain in a nutshell what the **test object** is.
* **Main focus:** Which areas should the tester primarily pay attention to?
* What **requirements** must be met in order to be able to test (e.g. device, OS, hardware, accounts, accesses/VPN, location, and what’s out of scope)?
* Avoid **technical terms and incomprehensible insider wording** (or at least include a glossary if it’s required).
* Put yourself in the position of a layman. Check yourself or a colleague to see **if your test scope is really understandable and all the necessary aspects** have been taken into account.

### 2. Make test cases that are understandable for everyone

Please note the following bullets when formulating test cases, especially when testing for external audiences or crowds:

* Choose a clear **title** that describes the test case.
* Describe the test case in **a short summary** to help the tester get started.
* Specify **a clear processing time** of the test case. This is extremely important if the test case has to be run without interruption within a certain time.
* If necessary, provide **background information** in each step. Describe exactly what the tester should expect in the next step and what effect this has.
* Identify which steps require **evidence** (screenshots, video, logs, etc) and add requirements for those.
* Be sure to name the **key functionalities** of the product that must work 100%.
* **Build the test case intuitively** and according to the product. Put yourself in the position of a user: What exactly does this usually do? Does the course of the test case reflect the subsequent user experience or user journey realistically?
* **Bundle all important test data** within the test case and assign the specific test data to the individual steps. This avoids confusion and prevents the tester from jumping back and forth between steps. Let the tester fully concentrate on the chronological execution of the test case.
* **Important:** The individual steps of the test case and the expected results must be written in a short and comprehensible way. Add one more step rather than overloading a single step with information unnecessarily.

### 3. Obtain feedback from the testers

Close contact with the testers and [**regular feedback sessions**](https://www.applause.com/blog/test-case-management-key-to-exceptional-qa) on new and existing test cases are therefore extremely important if you want to continually optimize your test cases. You may even want to include a “feedback form” in test cases so people can comment if they found something unclear.

### 4. Review of the test cases

Now that all steps have been carefully followed, the test run is over, the results of the test are available and you have received valuable feedback from the testers, you should take the time to discuss your results with all people involved in the testing. Take any kind of feedback and let it flow into the next testing cycle to save both time and money and learn from possible mistakes.

[**Quality Assurance is an ongoing process**](https://www.applause.com/blog/software-testing-life-cycle-stlc-phases)**.** Try to use each new test cycle to ultimately get the best version of the test object.

1. **What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

**Answer :** Software Testing Life Cycle (STLC) is the testing process that is executed in a well-planned manner. In the STLC process, various activities are carried out to improve the quality of the product. However, STLC phases only deal with testing and detecting errors but not development itself.

Different companies define different phases in STLC. However, generic Software Test Life Cycle has the following stages.

1. Requirement Analysis
2. Test Planning
3. Test Development
4. Test Environment Setup
5. Test Execution & Closure

ere, are Important reasons for using STLC method:

* STLC helps make the testing process more sophisticated, consistent and effective
* You can include milestones and deliverables for each step of the project
* Easy to understand and implement even if the model is expanded to various levels
* Time constraints are strongly built in project formulation
* Each module of the project is tested before the beginning of the another module
* The requirement of the specific project is measured against the actual result
* STLC analyze system requirements collected from clients and stakeholders
* Helps you to create traceability Matrix
* Identify the testing technique and testing types
* Prioritize the feature which should be primarily targeted on the test
* You can Analyze the Automation feasibility with STLC
* Identify the information about the testing environment where the actual test should be executed

**SDLC (Software Development Life Cycle)** defines all the standard phases which are involved during the software development process. SDLC life cycle is a process of developing software through a phased manner in the following order

1. Requirements Gathering
2. Design the software
3. Build the Software
4. Test
5. Deployment

|  |  |  |
| --- | --- | --- |
| **Parameter** | SDLC | STLC |
| Origin | Development Life Cycle | Testing Life Cycle |
| Objective | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. | The only objective of the STLC phase is testing. |
| Requirement Gathering | In SDLC the business analyst gathers the requirements and create Development Plan | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan |
| High & Low-Level Design | In SDLC, the development team creates the high and low-level design plans | In STLC, the test analyst creates the Integration Test Plan |
| Coding | The real code is developed, and actual work takes place as per the design documents. | The testing team prepares the test environment and executes them |
| Maintenance | SDLC phase also includes post-deployment supports and updates. | Testers, execute regression suits, usually automation scripts to check maintenance code deployed |

1. **What is the difference between test scenarios, test cases, and test script?**

**Answer :** Difference between test scenarios and test cases is that

Test Scenarios: A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.

Test Cases: It is a document that contains the steps that have to be executed; it has been planned earlier.

Test Script: It is written in a programming language and it's a short program used to test part of the functionality of the software system. In other words a written set of steps that should be performed manually.

1. **Explain what Test Plan is? What is the information that should be covered?**

**Answer :** A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”

Let’s start with following Test Plan example/scenario: In a meeting, you want to discuss the Test Plan with the team members, but they are not interested – .

## What is the Importance of Test Plan?

Making Test Plan document has multiple benefits

* Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
* Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
* Important aspects like test estimation, test scope,[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

## How to write a Test Plan

You already know that making a **Test Plan** is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables