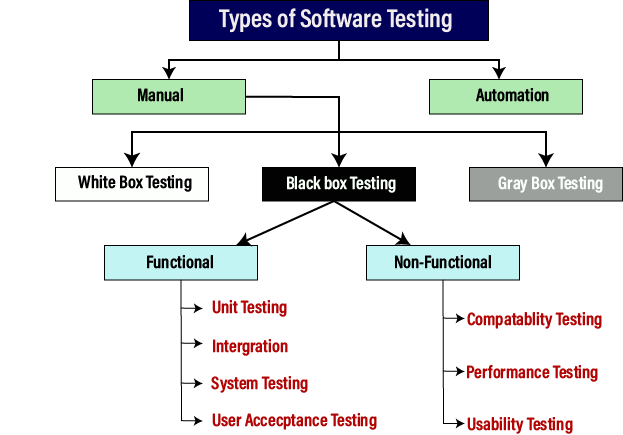
**Types of Software Testing**

The basic goals of software tests are to eliminate bugs and to enhance various aspects of the software, such as performance, user experience, security, and so on. A great deal of testing can amazingly improve the overall quality of the software, which will lead to great customer satisfaction.

But is software testing essential? What if we don’t do this?

Nowadays, software applications are used everywhere -- hospitals, traffic, shops, business organizations, etc. So not testing the software at all is dangerous. It’s dangerous in the sense that it can cause severe harm, such as security breaches, loss of money, and even deaths in some cases. Delivering or launching an application without testing it very well will cause many small or big problems for the users.

Types of Software Testing



1. Black box testing

Performed by the QA team of a company, black box testing is a testing technique that involves the checking of the application’s functionality without having any technical knowledge of the application, like the knowledge of the code’s logic, how the code works, knowledge of the internal structure, etc.

Generic steps of black box testing

* The black box test is based on the specification of requirements, so it is examined in the beginning.
* In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
* In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
* The fourth phase includes the execution of all test cases.
* In the fifth step, the tester compares the expected output against the actual output.
* In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

1. White box testing

Performed by the development team, white box testing is a testing method that requires a good understanding of the application’s code. It requires great knowledge of the app’s internal logic.

The white box testing contains various tests, which are as follows:

* Path testing
* Loop testing
* Condition testing
* Testing based on the memory perspective
* Test performance of the program

1. GreyBox Testing

Greybox testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a combination of black box and white box testing because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

WhiteBox Testing + BlackBox Testing = GreyBox Testing

GreyBox testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

Why GreyBox testing?

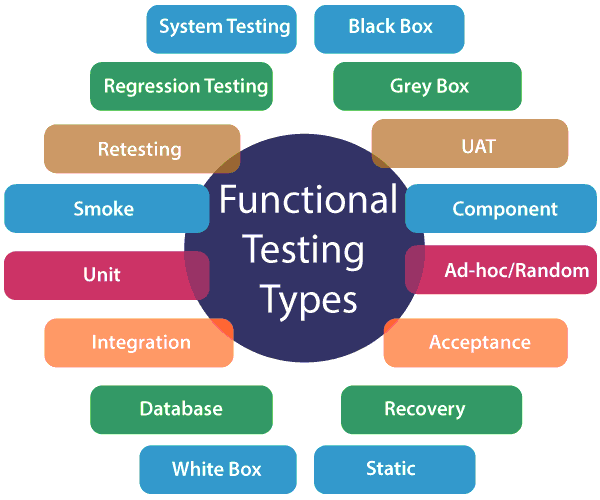
* Reasons for GreyBox testing are as follows
* It provides combined benefits of both Blackbox testing and WhiteBox testing.
* It includes the input values of both developers and testers at the same time to improve the overall quality of the product.
* It reduces time consumption of long process of functional and non-functional testing.
* It gives sufficient time to the developer to fix the product defects.
* It includes user point of view rather than designer or tester point of view.
* It involves examination of requirements and determination of specifications by user point of view deeply.

1. Functional Testing

Functional testing involves the testing of the functional aspects of a software application. When you’re performing functional tests, you have to test each and every functionality. You need to see whether you’re getting the desired results or not.

There are several types of functional testing, such as:

* Unit testing
* Integration testing
* End-to-end testing
* Smoke testing
* Sanity testing
* Regression testing
* Acceptance testing
* White box testing
* Black box testing
* Interface testing



Non-Functional Testing

* Load testing
* Failover testing
* Compatibility testing
* Usability testing
* Scalability testing
* Volume testing
* Stress testing
* Maintainability testing
* Compliance testing
* Efficiency testing
* Reliability testing
* Endurance testing
* Disaster recovery testing
* Localization testing
* Internationalization testing

**Different Types of Software Testing**

1. **Unit Testing**

Testing each component or module of your software project is known as unit testing. To perform this kind of testing, knowledge of programming is necessary. So only programmers do these kinds of tests, not testers.

You have to do a great deal of unit testing as you should test each unit of code in your project.

1. **Integration testing**

After integrating the modules, you need to see if the combined modules work together or not. This type of testing is known as integration testing. You need to perform fewer integration tests than unit tests.

Some good tools for unit and integration testing are [Jasmine](https://jasmine.github.io/), [Mocha](https://mochajs.org/), etc.

1. **End-to-end Testing**

End-to-end testing is the functional testing of the entire software system. When you test the complete software system, such testing is called end-to-end testing. You need to perform fewer end-to-end tests than integration test

[Cucumber](https://cucumber.io/), [Protractor](https://www.protractortest.org/), [Jasmine](https://jasmine.github.io/), [Karma](https://karma-runner.github.io/1.0/index.html), [SpecFlow,](https://specflow.org/community/talking-about-specflow/) etc. are some great end-to-end testing tools.

**4. User Interface Testing**

User interface testing involves the testing of the application’s user interface. The aim of UI tests is to check whether the user interfaces have been developed according to what is described in the requirements specifications document.

By running UI tests, you can make the application’s user interfaces more user-friendly and appealing to the eyes.

Some great automated user interface testing tools are [Monkey test for Android](https://developer.android.com/studio/test/monkey.html), [Saucelabs](https://saucelabs.com/), and [Protractor](http://www.protractortest.org/#/).

**5. Accessibility testing**

Testing whether your software is accessible to disabled people or not is termed as accessible testing. For this type of tests, you need to check if disabled people such as those who are color blind, blind, and deaf can use your application.

The right choice of color and contrast need to be made to make your software accessible to color-blind people.

**6. Alpha testing**

Alpha testing is a kind of testing to look for all the errors and issues in the entire software. This kind of test is done at the last phase of app development and is performed at the place of the developers, before launching the product or before delivering it to the client to ensure that the user/client gets an error-free software application.

Alpha testing is run before the beta testing, which means that after performing alpha testing, you need to run beta testing.

Alpha testing is not performed in the real environment. Rather, this kind of tests is done by creating a virtual environment that resembles a real environment.

**7. Beta testing**

As said earlier, beta testing takes place after alpha testing. Beta testing is done before the launch of the product. It is carried out in a real user environment by a limited number of actual customers or users, in order to be certain that the software is completely error-free and it functions smoothly. After collecting feedback and constructive criticism from those users, some changes are made to make the software better.

So when the software is under beta testing, it is called beta version of the software. After this testing is complete, the software is released to the public.

**8. Ad-hoc testing**

As the name suggests, ad-hoc testing is a kind of testing that is performed in an ad-hoc manner, without using any test cases, plans, documentation, or systems. Unlike all other types of testing, this kind of testing is not carried out in a systematic manner.

Although finding errors can be difficult without using test cases, there are technical issues that are easily detected through an ad-hoc test, but are hard to find through other testing approaches that use test cases.

This informal type of software testing can be executed by any person involved with the project.

**9. Compatibility testing**

Compatibility testing involves compatibility checking of the software with different operating systems, web browsers, network environments, hardware, and so on. It checks whether the developed software application is working fine with different configurations.

To give you a few examples, if the software is a Windows app, it should be checked whether it is compatible with different versions of the Windows operating system. If it’s a web application, it is tested whether the app is easily accessible from different versions of the widely-used web browsers. And if it’s an Android app, it should be checked whether it is working well with all the commonly used versions of the Android operating system.

**10. Backward compatibility testing**

Backward compatibility testing is carried out to test if a brand new or an updated version of an application is compatible with the previous versions of the environments (such as operating systems and web browsers) on which the software runs. Sometimes, some application is updated specifically to match the standard and style of a newer, more modern environment. In that case, support for backward compatibility is necessary.

Backward compatibility testing ensures that all those who are using the older versions of a particular environment can use your software.

**11. Browser compatibility testing**

As the name says, browser compatibility testing checks a web application for browser compatibility. More specifically, it tests whether the web app can easily be accessed from all versions of the major web browsers.

It is a specific form of compatibility testing, while compatibility testing checks for general compatibility.

Some popular tools to check browser compatibility include [CrossBrowserTesting.com](https://crossbrowsertesting.com/), LamdaTest, [Browsershots](http://browsershots.org/), [Experitest](https://experitest.com/), [Turbo Browser Sandbox](https://turbo.net/browsers), [Ranorex Studio](https://www.ranorex.com/cross-browser-testing-tools/), [Browsera](http://www.browsera.com/), etc.

**12. Performance testing**

Performance tests are run to check if the software’s performance is good or not. There are performance testing tools that analyze your app’s performance and show you the performance issues. By fixing those issues, you’ll be able to increase the performance of your software application.

Some great performance testing tools, also known as load testing tools, for web applications are [WebLOAD](https://www.radview.com/), [LoadView](https://www.loadview-testing.com/), [NeoLoad](https://www.neotys.com/), [LoadNinja](https://loadninja.com/), [Appvance](https://www.appvance.ai/), [LoadRunner](https://www.microfocus.com/en-us/products/loadrunner-load-testing/overview), [Apache JMeter](https://jmeter.apache.org/), [Loadster](https://loadster.app/), [LoadImpact](https://loadimpact.com/), [Testing Anywhere](https://testanywhere.co/), [SmartMeter.io](https://www.smartmeter.io/), [Tricentis Flood](https://www.tricentis.com/resources/getting-started-with-load-testing-flood-io/), [Rational Performance Tester](https://www.ibm.com/developerworks/downloads/r/rpt/index.html), [LoadComplete](https://support.smartbear.com/loadcomplete/docs/index.html), etc.

**13. Load testing**

Load testing is one kind of performance testing that tests how much load a system can take before the software performance begins to degrade. By running load tests, we can know the capacity of taking load of a system.

You can run load tests using tools like [LoadRunner](https://www.microfocus.com/en-us/products/loadrunner-load-testing/overview), [WebLoad](https://www.radview.com/), [JMeter](https://jmeter.apache.org/), etc.

**14. Recovery testing**

Recovery testing involves the checking of whether the application can recover from crashes and how well it recovers. In this kind of tests, testers observe how well the software can come back to the normal flow of execution. Crashes can happen anytime. Even if your software is of exceptional quality, crashes may happen. You don’t know when they may take place and annoy the users.

So you have to implement mechanisms that will recover the software application quickly and that will make the application run smoothly again.

**15. Regression testing**

If you need to make changes in any component, module, or function, you have to see if the whole system functions properly after those modifications. Testing of the whole system after such modifications is known as regression testing.

This type of testing concentrates on making sure that the code changes should not side-effect the existing functionality of the system. Regression testing specifies when a bug arises in the system after fixing the bug, regression testing concentrates on whether all parts are working or not. Regression testing focuses on whether there is any impact on the system.

**16. API testing**

Just like unit testing, API testing is also a code-level testing type. The basic difference between unit testing and API testing is that unit testing is performed by the development team whereas API testing is handled by the QA team.

**17. Security testing**

Security tests are performed to ensure the security of your application, in order that security breaches can be prevented. Security experts run this kind of tests to see how much your software is secure from attacks and to find security issues so that the app’s security can be strengthened.

The [top website security testing tools](https://hackr.io/blog/top-10-open-source-security-testing-tools-for-web-applications) include Grabber, [Arachni](http://www.arachni-scanner.com/), [Iron Wasp](https://ironwasp.org/), [Nogotofail](https://security.googleblog.com/2014/11/introducing-nogotofaila-network-traffic.html), [SQLMap](http://sqlmap.org/), [W3af](http://w3af.org/), [Wapiti](http://wapiti.sourceforge.net/), [Wfuzz](http://www.edge-security.com/wfuzz.php), [Zed Attack Proxy](https://www.zaproxy.org/), etc.

**18. Usability testing**

Testing the user-friendliness of an app is known as usability testing. It involves the checking of how much usable or user-friendly the app is. It is tested whether any user can easily use your software without getting stuck.

One of the best ways to test the usability of your software is to invite a few people to use your software. See if they can do certain things in your app without taking any help from you.

Take a look at these useful usability testing tools: [Optimizely](https://www.optimizely.com/), [Qualaroo](https://qualaroo.com/), [Crazy Egg](https://www.crazyegg.com/), [Usabilla](https://usabilla.com/), [Clicktale](https://www.clicktale.com/default.aspx), [Five Second Test](http://fivesecondtest.com/), [Chalkmark](https://www.optimalworkshop.com/chalkmark), [UXtweak](https://www.uxtweak.com/usability-testing/tools-and-software).

**19. Scalability testing**

Scalability testing verifies whether the software is scalable or not. In other words, it checks if your app performs well when the number of users, amount of data, or the number of transactions increases significantly. A software application that is not scalable may cause great business loss.

**20. Reliability testing**

Reliability testing is a type of software testing that verifies if the software is reliable or not. In other words, it checks whether the software runs error-free and that one can rely on it.

For example, if a user’s important information stored in the database of the software gets suddenly deleted after a few months because of some error in the code, we can say that the software is not reliable.

**21. Acceptance testing**

The client who will purchase your software will perform acceptance testing (also known as User Acceptance Testing) to see if the software can be accepted or not by checking whether your software meets all the client’s requirements and preferences. If your software doesn’t meet all the requirements or if your client doesn’t like something in the app, they may request you to make changes before accepting the project.

**22. Smoke Testing**

Smoke Testing is a software testing technique performed post software build to verify that the critical functionalities of software are working fine. It is executed before any detailed functional or regression tests are executed. The main purpose of smoke testing is to reject a software application with defects so that QA team does not waste time testing broken software application.

In smoke testing, the test cases chose to cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.  
For Example, a typical smoke test would be – Verify that the application launches successfully, Check that the GUI is responsive

Smoke Testing: Functional testing by smoke testing. Smoke testing includes only the basic (feature) functionality of the system. Smoke testing is known as "Build Verification Testing." Smoke testing aims to ensure that the most important function work.

For example, Smoke testing verifies that the application launches successfully will check that GUI is responsive.

**23. Sanity Testing**

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

Sanity Testing: Sanity testing involves the entire high-level business scenario is working correctly. Sanity testing is done to check the functionality/bugs fixed. Sanity testing is little advance than smoke testing.

Integration, Functional UI Test, Smoke/Sanity checks, Regression/retest, performance test beside automation test