Types of Software Testing

Software testing is generally classified into two main broad categories: functional testing and non-functional testing. There is also another general type of testing called maintenance testing.

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

Functional tests are performed both manually and using automation tools. For this kind of testing, manual testing is easy, but you should use tools when necessary.

Some tools that you can use for functional testing are [Micro Focus UFT](https://www.microfocus.com/en-us/products/unified-functional-automated-testing/overview) (previously known as QTP, and UFT stands for Unified Functional Testing), [Selenium](https://www.seleniumhq.org/), [JUnit](https://junit.org/junit5/), [soapUI](https://www.soapui.org/), [Watir](http://watir.com/), etc.

2. Non-functional Testing

Non-functional testing is the testing of non-functional aspects of an application, such as performance, reliability, usability, security, and so on. Non-functional tests are performed after the functional tests.

With non-functional testing, you can improve your software’s quality to a great extent. Functional tests also improve the quality, but with non-functional tests, you have the opportunity to make your software even better. Non-functional testing allows you to polish the software. This kind of testing is not about whether the software works or not. Rather, it’s about how well the software runs, and many other things.

Non-functional tests are not generally run manually. In fact, it’s difficult to perform this kind of tests manually. So these tests are usually executed using tools.

There are several types of non-functional testing, such as:

* Performance testing
* Security 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

Note that explaining all the types of software testing is beyond the scope of this article.

Different Types of Software Testing

This article explains only some of the most common 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 this kind of tests, not testers.

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

2. 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.

3. 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 tests.

[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 is tested 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.

### 16. Agile testing

Carried out by the QA team, Agile testing is a type of testing that is conducted according to the rules of agile methodology. This kind of testing is done from the actual customers’ viewpoint.

Some useful tools that you can use for Agile testing are [JIRA](https://www.atlassian.com/software/jira), [PractiTest](https://www.practitest.com/), [JunoOne](https://juno.one/), [VersionOne](https://www.versionone.com/versionone-vs-atlassian-jira-agile/), [TestRail](https://www.gurock.com/testrail), [SoapUI](https://www.soapui.org/), etc.

### 17. 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.

### 18. 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.

### 19. 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.

### 20. 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.

### 21. 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).

### 22. 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.

### 23. 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.

### 24. 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.

## ****Final words****

This article explained several types of software testing. Keep in mind that you don’t need to perform all of these tests mentioned in this post for your software project. What kinds of tests you should run depends on the type of software you’re building and other factors.

Besides performing tests, measuring the effectiveness of the tests is also important, and test coverage tells the effectiveness of your tests. [Istanbul](https://istanbul.js.org/) is a good tool for measuring test coverage, used for JavaScript software projects.

There can be undetected errors in your application even after it’s launched, which will annoy the users and will cause problems for them. Real-time error-checking tools such as [Sentry](https://sentry.io/welcome/) and [Newrelic](https://newrelic.com/) will automatically find errors and notify you, so you don’t need to tell your users to report bugs.

You can also use automated code grading tools. Automated code grading tools like [sonarqube](https://www.sonarqube.org/) and [codebeat](https://codebeat.co/) help you amazingly improve the quality of your code by showing issues in your application. These tools will help you fix bugs in less time. After analyzing your code, these tools give you useful reports with valuable information required for code quality enhancement.

You can use programs called linters to check if the code of your software project meets the specified coding convention rules. A linter actually saves you a lot of time as manually checking the code written by several developers is a very time-consuming process.

You can find linters for almost any programming language. Take a look at these popular linters: [TypeScript TSlint,](https://www.npmjs.com/package/tslint)[JavaScript ESLint](http://eslint.org/), [Sass/SCSS sass-lint](http://sass-lang.com/), [Python pylint](https://www.pylint.org/)/[flake8](http://flake8.pycqa.org/en/latest/), [Bash ShellCheck](https://www.shellcheck.net/), [Go golang lint](https://github.com/golang/lint), etc. Code editors such as Visual Studio Code let you configure linting.