# Software Testing Tutorial: A Comprehensive Guide With Examples And Best Practices

Software testing is the process of verifying the working of the software system or application. In other words, it ensures the software application is bug-free and addresses the technical and user requirements. It not only focuses on finding bugs or errors in the software but also considers measures to enhance the software quality in terms of usability, accuracy, and efficiency.

Suppose you are building a software application that involves writing a bunch of codes and fixing bugs. These are part of the Software Development Life Cycle that developers and testers follow. However, it is their accountability to check or verify the code in line with the requirements and performance of the application.

But what happens when an application is riddled with bugs that impact its features and quality? How could this be eliminated? Here enters software testing, a highly crucial part of the Software Development Life Cycle.

**What Is Testing?**

Testing can be understood as the technique used to determine the quality of the application based on the Software Requirement Specification (SRS). However, it is not possible for testing to detect all defects in an application. Testing aims to find the reason behind the failure of an application so it can be resolved.

Some of the critical points on testing include:

* Provide a comparison between the state and working of software applications against different test scenarios.
* Involves validation and execution of test scripts in different environments.
* Gives detailed test reports on the errors and bugs.

Testing is essential, as it involves the risk of application failure without testing. Hence, an application cannot be deployed to the end user without software testing.

Now, let us connect with software testing.

**Introduction to Software Testing**

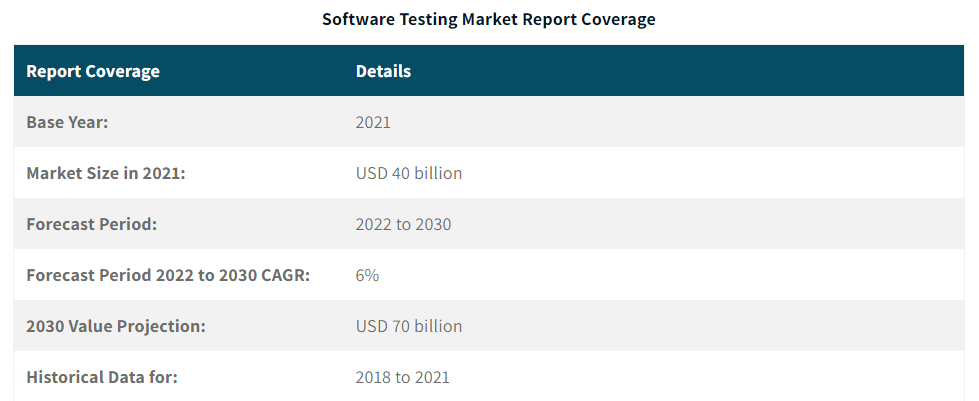
Software testing is the process of assessing the functionality of a software application. It checks for errors or bugs by evaluating the execution of the software’s components to meet the needs of the expected outcome. It identifies the accuracy and appropriateness of the application by taking into account its key attributes.

Some of those attributes include:

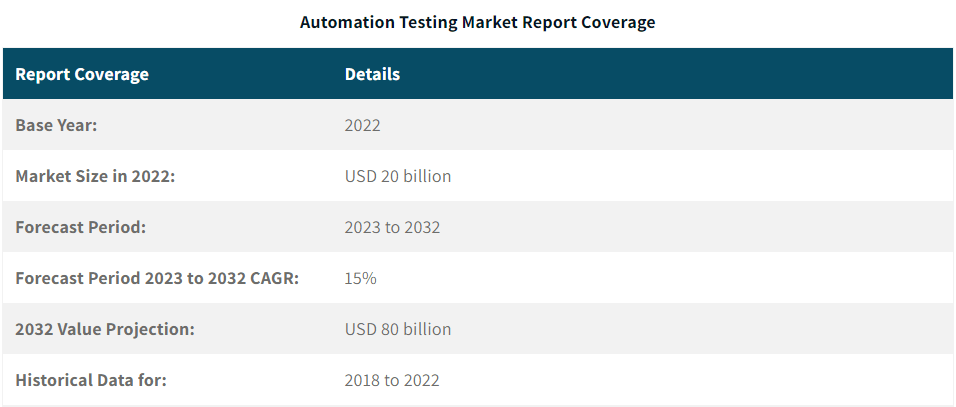
* Reliability
* Scalability
* Portability
* Reusability
* Authentic
* Practical
* Usability

It tends to provide an autonomous perception of software while ensuring its quality and hence is marked as the goal of software testing. To accomplish this, you test the software from a single line of code, its block, or even when the application is completed. Therefore, software testing holds specific objectives that allow better integration into the Software Development Life Cycle.

Software testing in the present market is gaining immense popularity. According to [Global Market insight](https://www.gminsights.com/industry-analysis/software-testing-market), the size of the software testing market in 2019 was $40 billion and is anticipated to surge at a CAGR of 6% in 2026. The increasing demand for automating software development allows a surge in industry share.



Additionally, as per Global Market insight, the automation testing market size in 2022 was valued at USD 20 billion and is expected to increase by over 15% CAGR within 2032.



Some of the underlying objectives that software testing holds are:

* Investigate the defects and ensure the product's performance as per the specification.
* Gives surety to the product that it meets the market standards.
* Solves any of the challenges or loopholes at the production stage.
* Eliminate future failure of the product.

It can be said that software testing should be carried out with a systematic approach to find defects in software. We know that technology is advancing and things are getting digitized. It is now easy to access online bank accounts, shop online from home, and have endless options.

However, what if these systems turn out to be malfunctioning? A single defect can cause substantial financial losses for the organization. This is the main reason for the tremendous rise of software testing and holding a solid grip on IT.

It is quite a common situation where a product experiences some defect; however, design errors may cause trouble if they go unnoticed. Hence, testing software applications is needed to ensure software development meets user requirements.

Have you wondered if the software is deployed with bugs embedded? It will make error detection very tough for the testers. The reason is that screening thousands of lines of code and then fixing the errors is a huge problem. It can also happen that fixing one error may unknowingly lead to the rise of another bug in the system.

Thus, testing software applications is characterized as a very important and essential part of software development. It is recommended to include software testing in every stage of the Software Development Life Cycle.

Let's sum up software testing from the below-given points:

* Testing software applications are needed to investigate the software's credibility.
* Ensures a bug-free system, which can lead to failure.
* Ensures software is related to the needs of the user.
* Ensures the final product is user-friendly.
* Software is developed by humans and is prone to error. Developing software with no defects is impossible without including software testing in the development cycle.

In the below section, we will get more clarity on the need for software testing.

**Why Do You Need Software Testing?**

The occurrence of defects in software can be due to many reasons. But, the matter of fact is that not all defects could pose threats to the system. We can accomplish much with software testing to ensure the software's effectiveness and quality. For example, any severe defect can delay the timely release of the product, which can lead to financial loss. Testing software applications is needed as it lowers the overall cost of the software development cycle.

If, in any case, software testing is not executed in the initial stage of software development, it may turn out to be highly expensive later. Monetary and human losses are the noted consequences, and history is full of such examples.

* In April 2015 [Bloomberg terminal in London](https://www.cnbc.com/2015/04/17/bloomberg-trading-terminals-experience-outage.html)crashed because of a software malfunction. It affected the lives of more than 300,000 traders.
* Failure of the POS system caused the [shutdown of 60%](https://www.cbsnews.com/news/starbucks-breakdown-shows-how-registers-have-evolved/)of Starbucks stores in the US.
* Software failure noted in the sensory airbag detector made [Nissan cars recall](https://www.usatoday.com/story/money/cars/2014/03/26/nissan-recall-air-bags-altima-sentra-leaf/6902937/) over 1 million cars from the market.

Such losses are based on the fact that tracking back to find the bug or defect is a huge challenge. As a result, software testing aids in preventing the emergence of such scenarios.

Software testing is required because it ensures that the software works and looks exactly like the specifications. Therefore, it strengthens the market reputation of an organization.However, you should also know when software testing should be executed.

Some specific needs for testing software applications are summarized in the following points:

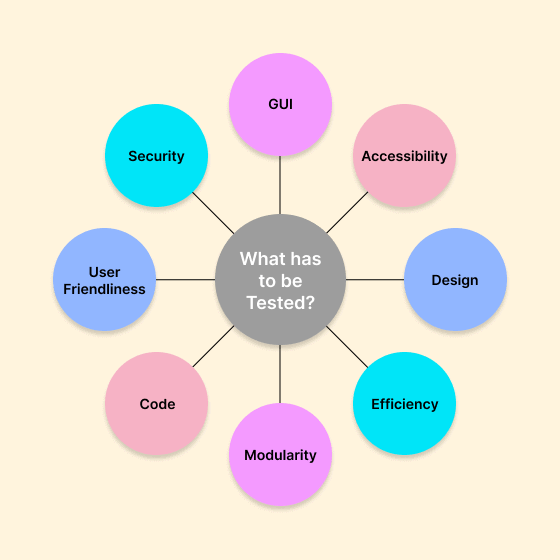
* Help identify bugs within written code.
* Improvise product quality.
* Gain customer confidence.
* Cut huge costs.
* Optimize business.
* Speeds up software development.

**What to Analyze in Software Testing?**

Having good information and an understanding of the project's needs is something a tester should be aware of. A good idea about the real-time environment where the software will be executed allows the tester to perform software testing efficiently. Hence, it is equally crucial for them to know what needs to be tested to devise a testing strategy.

Below are the aspects that require software testing:

* Modularity
* Efficiency
* Design
* Accessibility
* GUI
* Code
* User-friendliness
* Security



In addressing the need for testing software applications, it is vital to get an idea of their significance. It will help you learn the criticality of testing software applications. Let us see its significance.

**Significance of Software Testing**

Software testing is significant in determining bug-free software. It involves validating the system components manually or using different test tools to analyze specific defects. Here, we highlighted crucial reasons why testing software application is essential:

**Ensure Product Quality**

It checks the quality of the product by following its requirements. It ensures the functionality of the product to give a flawless user experience. For example, if you plan to launch a web application, ensuring compatibility with different browsers, real devices, and operating systems is important. Here, software testing plays a crucial role in checking cross-browser compatibility using a cloud-based platform.

**Satisfying Customer Demand**

The main objective of software organizations is to ensure customer satisfaction with the software. For this, software testing is opted to offer a perfect user experience. It provides the user's trust and enhances the organization's reputation.

**Improve Development Process**

With the help of quality assurance, it is possible to search for different arrays of scenarios and errors to reproduce the error. For developers, it is easy to fix those errors in no time. Further, the testers work with the development team, accelerating the development process.

**The Addition of Features Becomes Easy**

It is often difficult to change the code of the application as lines of code are interlinked. Change in any one could affect the other and lead to a rise of bugs. However, software testing counteracts this issue as it will help you know the codebase's exact loophole. Thus, it helps the developer confidently add new software features.

**Define Software Performance**

Software testing maintains an application's performance by regularly checking for errors or bugs. If noted, it gets modified or corrected in real time.

**Ensure Security**

Software security is a priority of any organization. Breaking the application's security can lead to a breach of data and the risk of being in huge loss. Poorly tested applications could increase the risk of the vulnerability of being hacked. Therefore, users prefer well-tested applications as they guarantee their data's security.

**Allows Saving Cost**

It ensures the project's cost-effectiveness by allowing early detection of errors or bugs. Such bugs or errors can be easily rectified at an early phase at a very low cost. Therefore, it is important to get testing done without any delay.

**Core Concepts of Software Testing**

Testing software applications may be problematic for some testers. To avoid the underlying issues, like difficulty in identifying bugs, can be overcome by knowing the fundamental concepts. When you start testing, following the given fundamental concepts will help you along the way:

**Test Strategy**

Being a tester requires being effective with the test strategy, which lays out the picture of software testing. A test strategy will help you know what type of testing will be appropriate for your application. Further, it gives information on which test to execute, the required time, and the effort to make testing effective.

**Tip:** If you are setting up your first test strategy, you should focus on features that are priorities for your application.

**Test Plan**

When starting testing, a test plan is a must. It is a comprehensive document that entails test strategies, estimation, deadline, objective, and resources to complete the testing process.

You can take the testing plan as the path that will clarify your scope of testing, like what is tested, by whom, and for how long. Along with that, it also contains information on any dependencies.

**Tip:** You have to regularly update the testing plan as you find new bugs in the software application.

**Test Cases**

Test cases are defined as the set of actions executed on a system to determine whether it satisfies the requirements and functions of the software. Hence, it is written at a pace similar to the program. For example, if you sign into an account for an application, it is expected that you open the home dashboard every time. To execute this test, incorporate this information as a test case.

**Tip:** Always give value to the most critical part of the application when setting up test cases.

**Test Data**

Test data is significant in case you want to run tests on the data of real users. Some examples of test data include product orders, sets of names, and other pertinent information on the application. No developer wishes to delete or update a real set of data from a real user’s application. Hence, it is crucial to retain a set of test data that can be further modified to ensure each of the functions of the software application is effectively working.

**Tip:** Test data development should be done concurrently with test case development.

**Test Environment**

Testing software applications are equally important as the test environment in which it is performed. It is crucial to conduct tests on different devices, browsers, and OS to ensure software compatibility. For example, if you plan to execute performance or usability testing, you need to include various devices to execute the test on the application.

**Tip:** Always set up a test environment before initiating testing.

**When to Perform Software Testing?**

Being a tester, you often try to avoid any complexity in testing software applications. It is highly preferred that the testing team starts software testing earlier, as it would allow the developer to finish the development process on time. Further, it will also take time and cost.

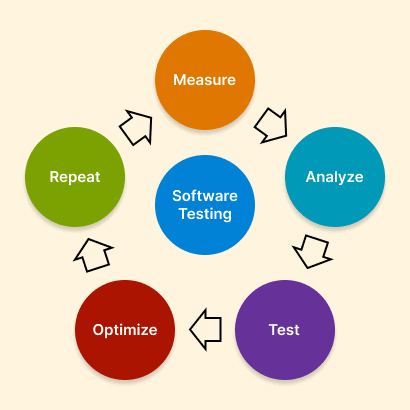
However, if you start testing at a later stage of the software development process, it might delay the software release and can turn out to be expensive. The main reason is the challenge of keeping track of the changes or bugs for rectification once the software reaches its final release phase.

Therefore, dividing the software development process into different phases is best. Then, perform testing in every such phase before heading forward to the next phase. It will allow you to quickly complete the software development process with adequate outcomes.

Additionally, it enables the integration of diverse modules, as you will be aware that independent testing executed for every module is done and working according to the given specification. But you might think about how much time testing can be done. It just depends on the project’s needs.

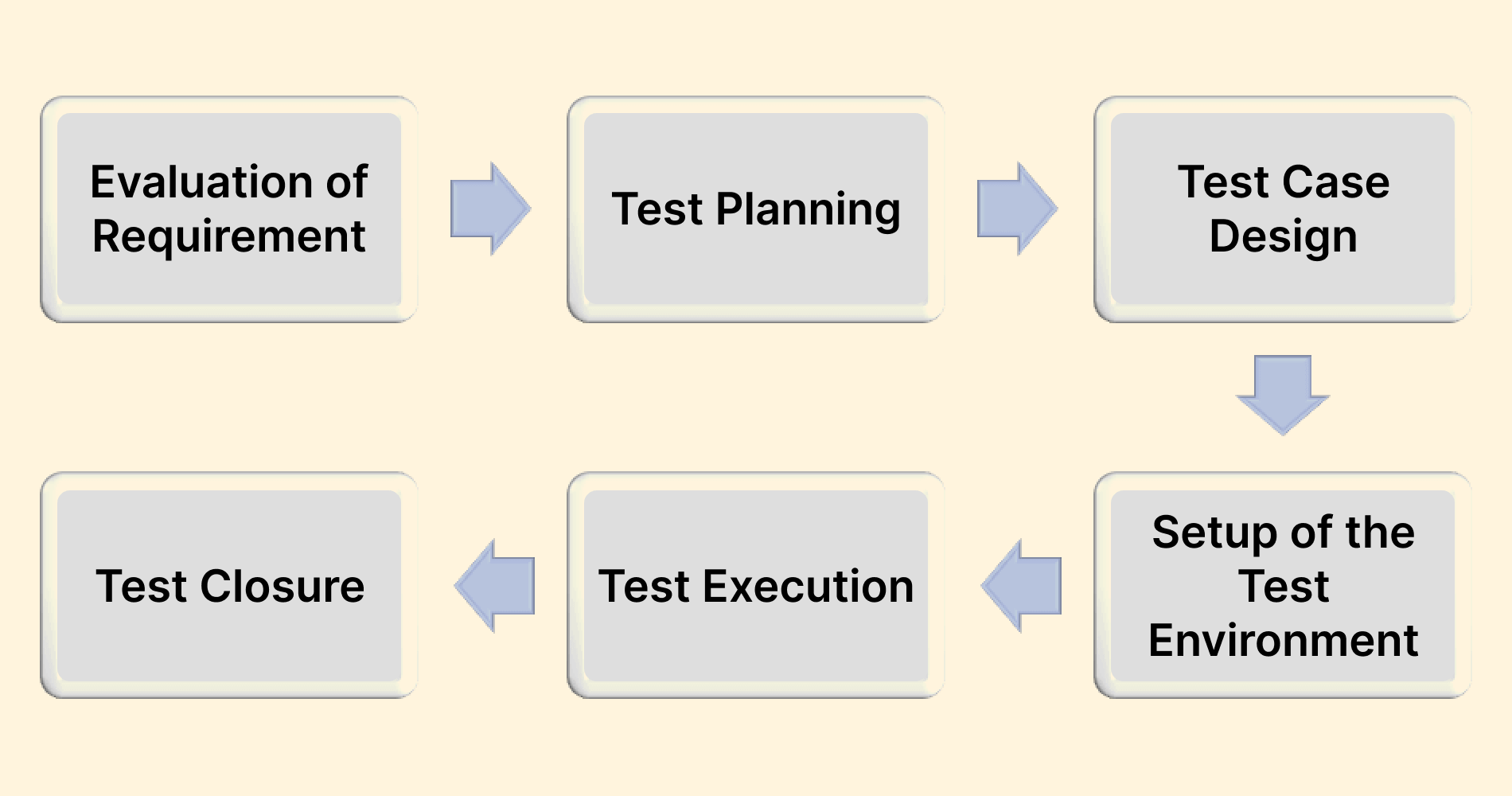
Also, software testing is not limited to any number. The frequency of software testing depends on how crucial the quality and security of the software application are to you and the organization. Preferably, testing needs to go in accordance with the development.

In such a process, testers are accountable for finding a maximum number of defects in the initial phase of software development. It is essential because, for example, if any bug fixes or modifications are required in the software application's design, they can be incorporated early.



**Phases of Software Testing**

Each software undergoes different phases of the Software Testing Life Cycle (STLC). It is the sequence of actions performed during the testing process to meet the quality of the software.



Detailed information on the different phases of software testing includes the following:

**Evaluation of Requirement**

It is the first phase in which the testing team identifies the testing requirements and items to be tested. The team defines such requirements (functional and non-functional) to check whether they are testable.

Actions required in the requirement analysis phase:

* Recognize specific tests to be done.
* Collect details on testing priorities and focus.
* Develop a Requirement Traceability Matrix (RTM).
* Recognize the test environment.
* Analysis of automation feasibility.

**Test Planning**

In the next phase, the testing team prepares a plan that defines the project's time, cost, and effort. In addition to this, factors like the test environment, test limitations, test schedule, and resources are also determined.

It includes the following elements:

* Developing a test plan and strategy
* Selection of testing tools
* Estimation of efforts
* Training needs

**Test Case Design**

Moving on to the test plan, testers work on writing and creating test cases. Identification of test data is followed by reviewing and reworking the test scripts and test cases. It includes the following action:

* Create test cases
* Review of test cases
* Create test data

**Setup of the Test Environment**

In the next phase, the testers determine the workings of the hardware and software of the products. It is usually done along with the test case development phase. Here, testers can do smoke testing for the given environment.

The following activities are included in this phase:

* Comprehend the environment set-up and architecture.
* Prepare test data and test environment.
* Conduct smoke tests on the build.

**Test Execution**

During the test execution phase, testers evaluate the software they created using test plans and test cases. It is done for test script maintenance, bug reporting, and test script execution. The included activities are:

* Run the test as per the test plan.
* Document test outcome.
* Locate failed test cases.
* Retest the failed cases.

**Test Closure**

It is the last phase of testing. It includes an effective exchange of information on testing artifacts to recognize strategies that can be executed in the future. The following activities are involved in the test closure phase:

* Test completion reporting.
* Collection of test completion metrics.
* Develop test closure reports.
* Reporting of quality of products.
* Analysis of test results.

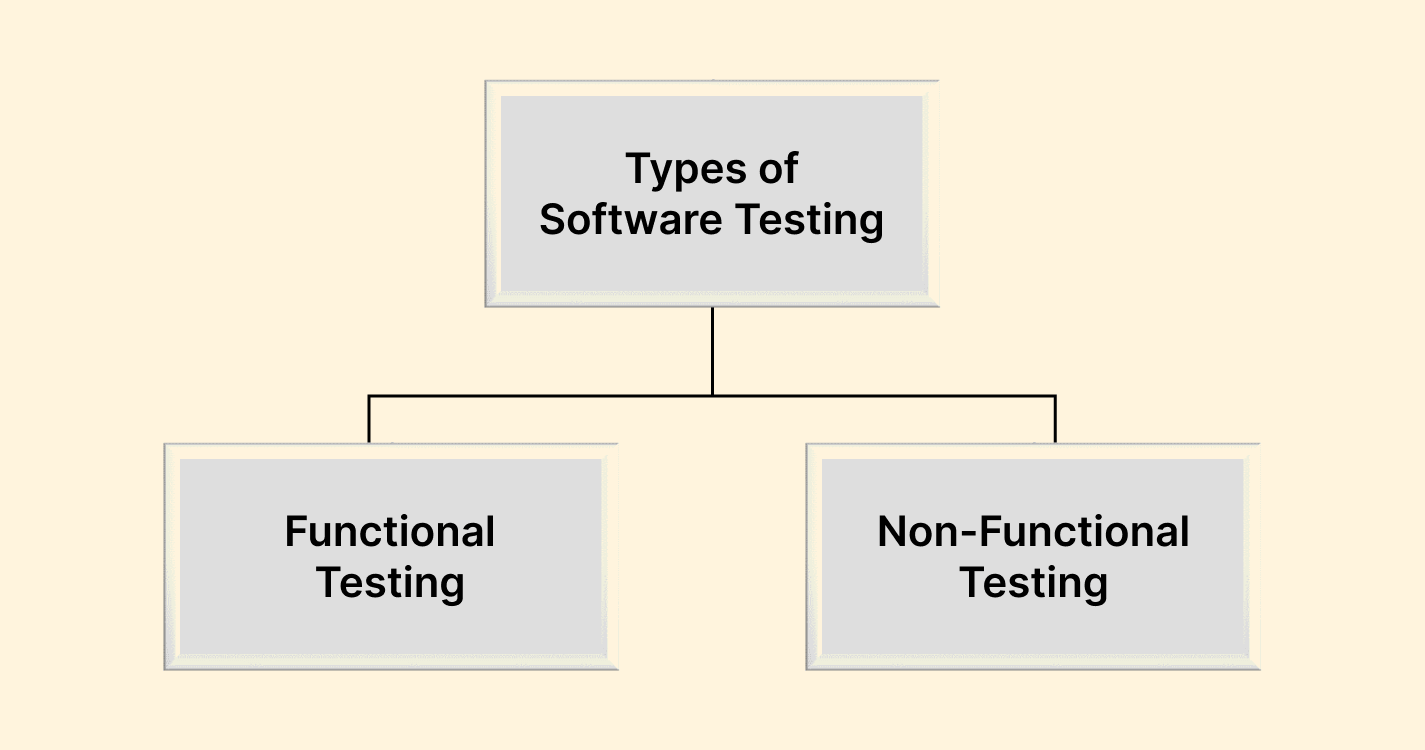
**Types of Software Testing**

In this section, we will discuss and go over various types of software testing in depth. In the software development cycle, the testers are often required to validate the software at different levels.

You must be aware of some common testing types, like functional testing, Agile testing, and others. To that end, various types of software testing provide a framework whose ultimate goal is to ensure the software application is bug-free.

Let's look at each type of testing.

Software testing is broadly classified as functional testing and non-functional testing. To get a better idea, look at the chart below.



**Functional Testing**

This type of testing considers the functional requirements of an application. It involves various actions and functions of the system for testing. This is achieved by providing input and comparing the actual output to the expected output. Here, the test cases prepared are according to the requirements of the software and the customer.

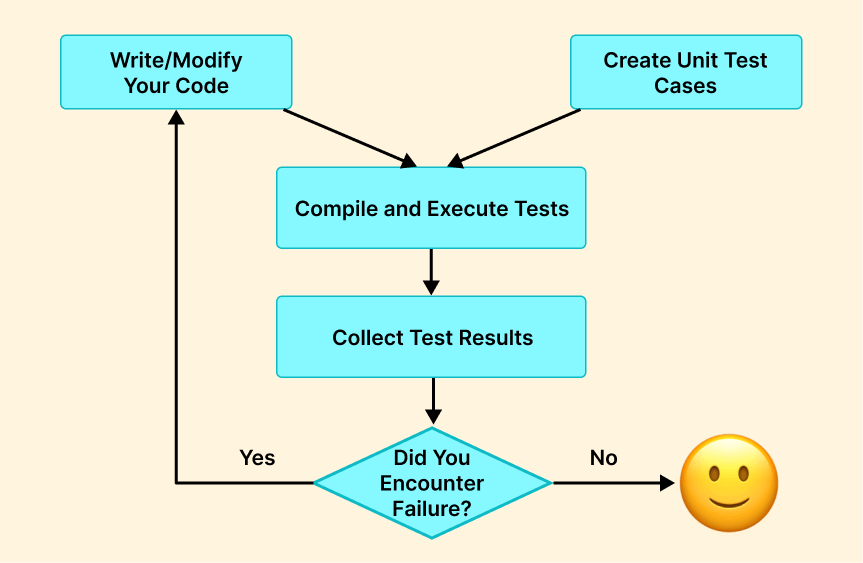
Generally, functional testing allows for the following checks:

* Testers should know the application's functionality.
* Always include the right set of data.
* The functional requirement of the application must align with the test data of the output application.
* All the test scenarios should be included.
* For the provided output, the result should be screened and recorded against the expected output.

Different types of functional testing include the following:

**Unit Testing**

Testing done on specific units and components of software can be understood as unit testing. Here, the verification of the individual units or part of the source code is tested. It is mainly executed at the early stage of software development.



A unit test case can be like clicking a button on a web page and validating that it works as expected. It is seen as a function, procedure, or method. For example, in unit testing, you will test the work of the login button to ensure it can route to the correct page link. To accomplish this, the developer mainly relies on automation testing tools for executing tests.

Unit testing includes two major types:

1. White Box Testing: It is also known as glass box or transparent testing. In white box testing technique, you test the internal structure or its underlying code base of the application. With this, it is easy to find any defect or error in the application's design. It follows techniques like data flow testing, path testing, decision coverage, and control flow testing.
2. Gorilla Testing: In this type of testing, inputs are repeatedly applied on a module to ensure appropriate functioning and bug-free application. It considers every piece of code and tests using random input unless it crashes. It checks the robustness of the application as it involves every module in the software application. Hence, due to its nature, it is also known as fault tolerance or torture testing

**Integration Testing**

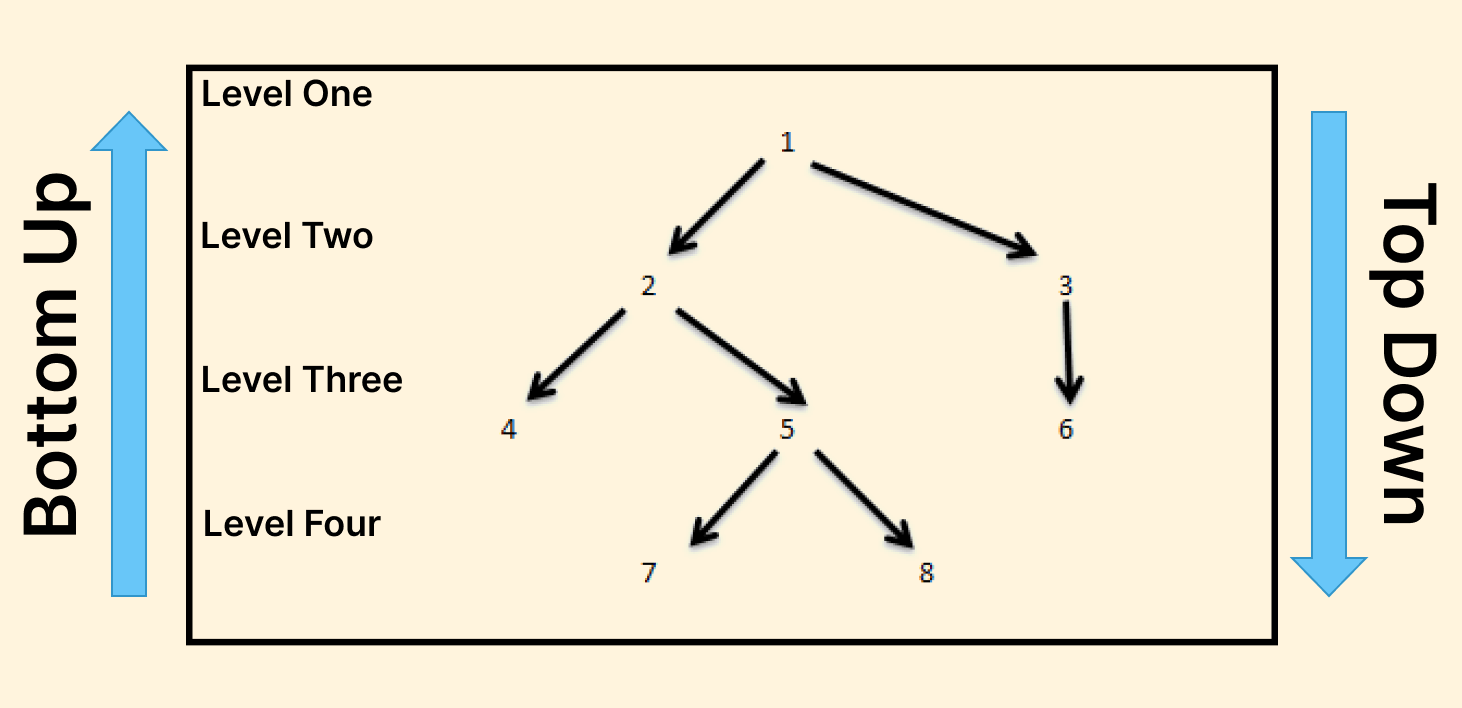
In this testing, different unit testing types are integrated to test the system as a whole. In other words, the application’s two or more two modules are integrated and tested.

The main aim of integration testing is to find bugs in the interface, data flow, and interaction among the modules. The testers investigate how different units associate and give output for different scenarios.

With the help of integration testing, errors about performance, requirements, and functional level are investigated. In unit testing, individual units are tested to know their performance as per expectation; however, in integration testing, such units' performance is checked when they are integrated.

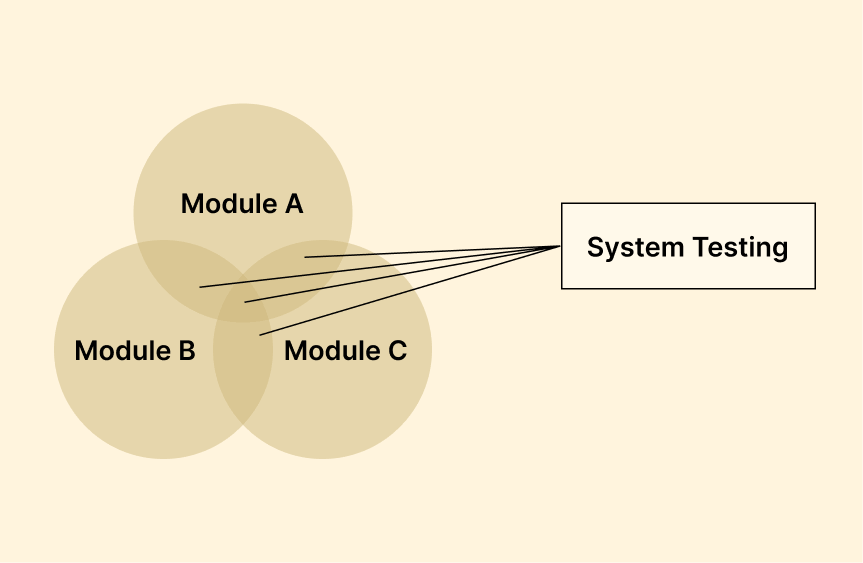
Integration testing is mainly classified into three types:

1. Big Bang: Here, all the application modules are integrated and combined to establish a complete system. Followed to this, testing for the bug is done.
2. Top-Down Approach: Initially, the top level of the module is tested, and then the addition of the sub-module is followed by its testing.
3. Bottom-Up Approach: It is contrary to the top-down approach. In such testing, a test for the lowermost modules is done. Followed to this, a stepwise approach is applied to add and test high-level modules.



**System Testing**

In system testing, all the integrated modules of the complete system are tested. It enables testers to verify and validate whether or not the system's requirements are met. It involves a different test that includes validating output in terms of particular input and the user's experience. Here, performance and quality standards are tested in compliance with the technical and functional specifications.

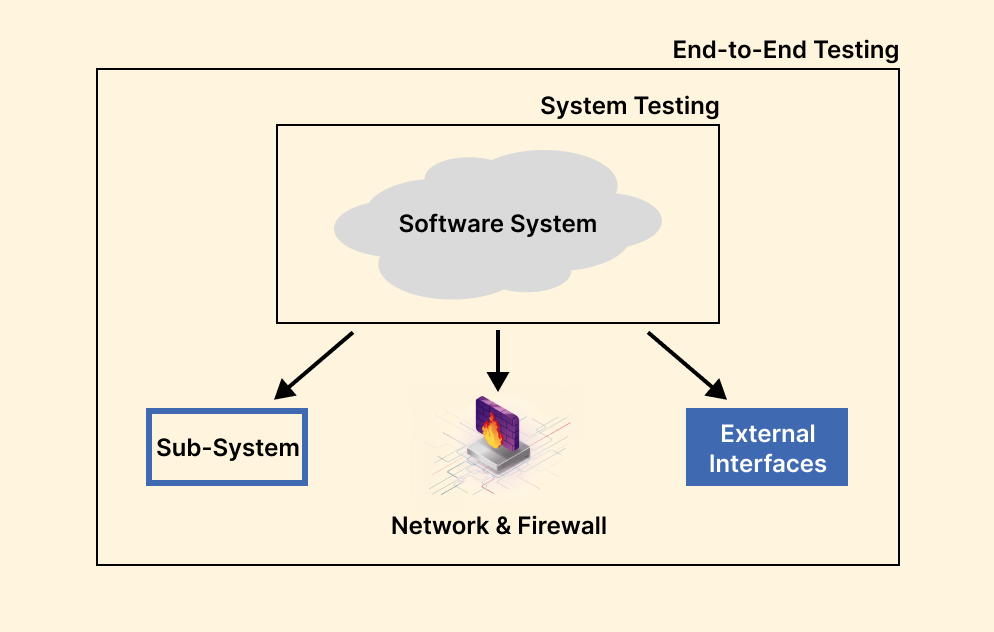


System testing is highly crucial when the system is deployed as it allows to development of a scenario resembling a real-time scenario. Hence, testing is mainly executed to investigate the behavior of the application, the architecture of the application, and the design of the software.

It includes many different software testing categories, which tend to verify the whole system. Some of those system tests include the following:

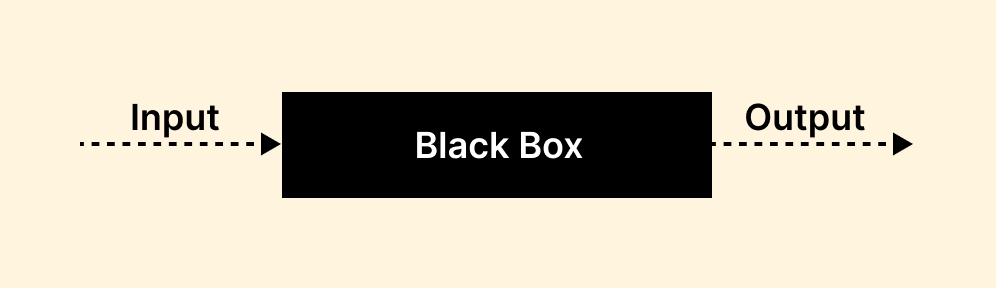
**End-to-End (E2E) Testing**

As the name suggests, end-to-end testing involves verification and validation of the workflow of the software applications from start to end. This type of testing aims to mimic or stimulate the real user scenario to verify the system for data integrity and integration. The main aim of end-to-end testing is to check the application as a whole for its dependencies, communication, data integrity, and interfaces to perform a complete production-like environment.



**Black Box Testing**

It is the testing method where the software application's internal code structure is unknown, and testing is done to verify its functionalities. Therefore, the main root of information in black box testing is the requirement specified by the customer.



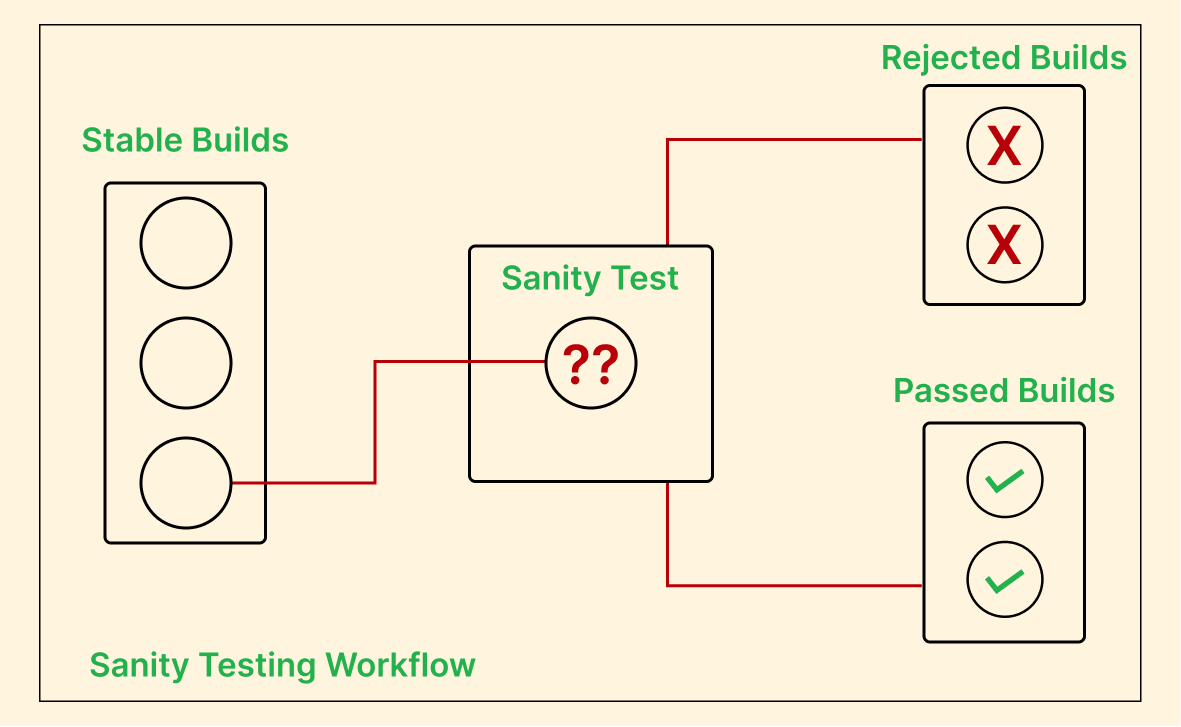
The QA team chooses a specific function and provides input value to validate its functionality in the software. It verifies whether the function provides an anticipated output or not. If the function does not provide the correct output, the testing is marked as failed; otherwise, it passes.

**Smoke Testing**

Smoke testing is also known as build verification testing. It aims to validate whether the basic and complex function of the system is working as expected. In other words, it helps to determine whether to build provided by the developers for deployed software pertains to stability. It ascertains to explain the significant function of the program; however, it does not investigate its finer details.

**Sanity Testing**

It is part of regression testing, where testers perform sanity testing to check whether the code changes done in the software are working or functioning as expected. Sanity testing is done in the system to validate that new changes, newly added functions, or the fixation of bugs do not lead to failure. Hence, the tester performs sanity testing to ensure a stable build.

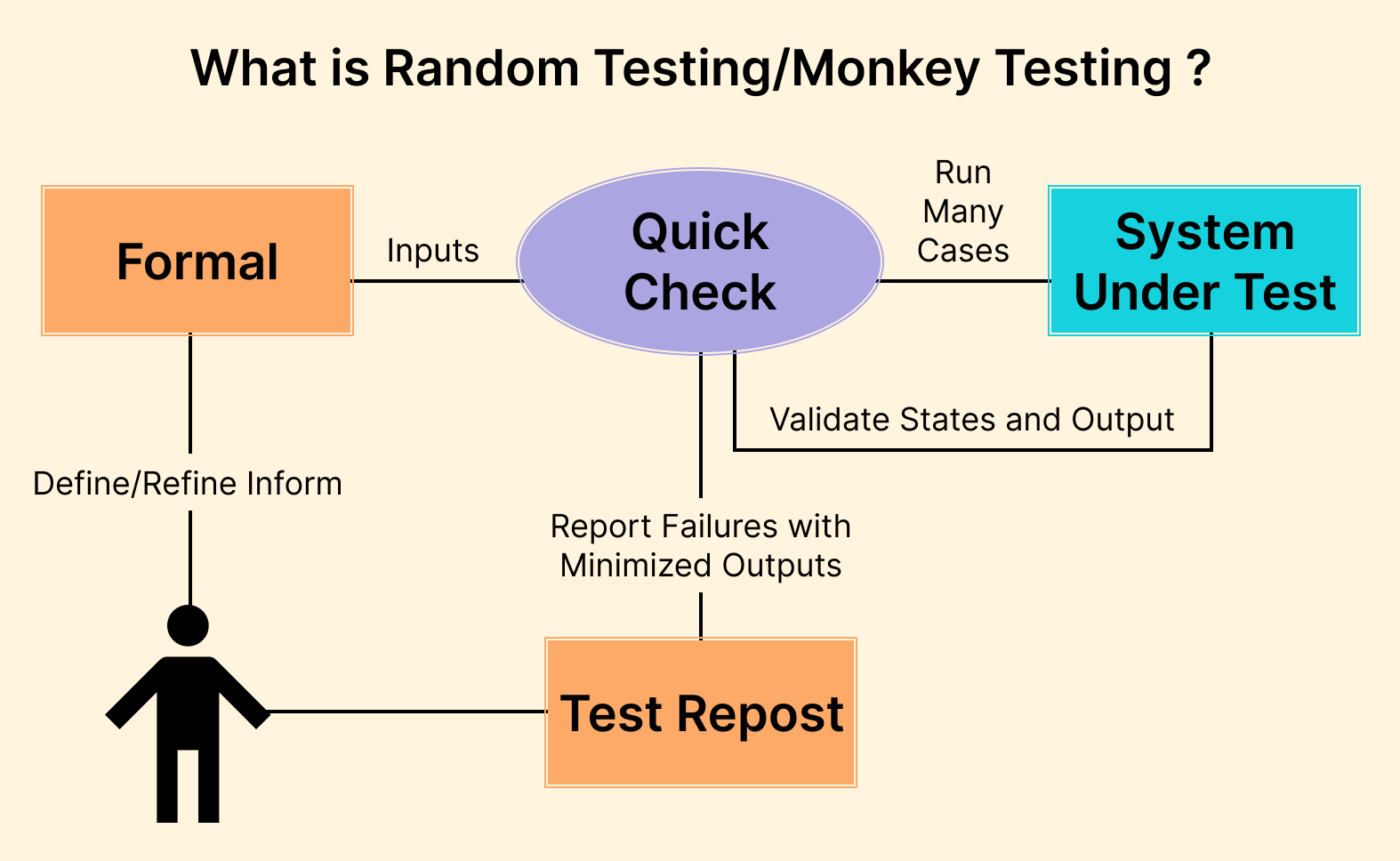


**Happy Path Testing**

It is also known as the golden path or sunny day testing. Happy path testing utilizes input and provides expected output. It tests the software application on a positive flow and does not consider errors or negative factors. The main focus area of this test is on positive input using, from which the application forms the expected output.

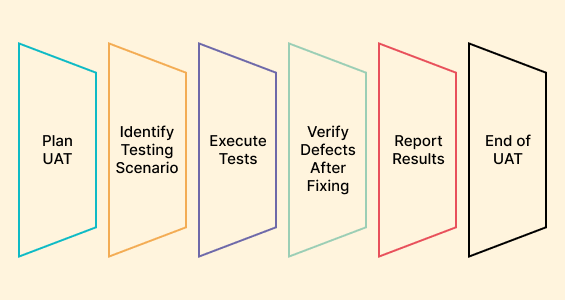
**Monkey Testing**

It is one of the types of system testing where random inputs of the application are provided by testers irrespective of any of the test cases to validate its behavior. Its main objective is to find new bugs to streamline the function of the application. The tester performs the test randomly without understanding the application's code. Hence, it is called monkey testing.



**Acceptance Testing**

In software application testing, on completion of unit, integration, and system testing, the next step is ensuring the application's quality. The QA team runs the test to define the quality in terms of predefined test cases and scenarios. In acceptance testing, the QA team investigates the whole system from the design view to its internal functions. It is marked to be very crucial in testing software apps as it considers the contractual and legal requirements of the application defined by the clients.



Acceptance testing includes many different types of testing. Some of the included types are explained below:

**Alpha Testing**

The quality and engineering aspects of the developed software are considered in alpha testing. In other terms, it tests the product in line with the business requirement. Hence, it allows ensuring the successful working of the product.

**Beta Testing**

It is performed at the end stage of the Software Development Life Cycle after completing the alpha testing. Beta testing is conducted in the real environment before the deployment of the product. Hence, it gives a report on the fact that there is no major software failure, and it satisfies the end user's needs.

**User Acceptance Testing**

Before deploying the software application, the users test it. In user acceptance testing, specific requirements of the software application users use are primarily chosen for testing. Hence, it is also known as "end-user testing.".

**Operational Acceptance Testing**

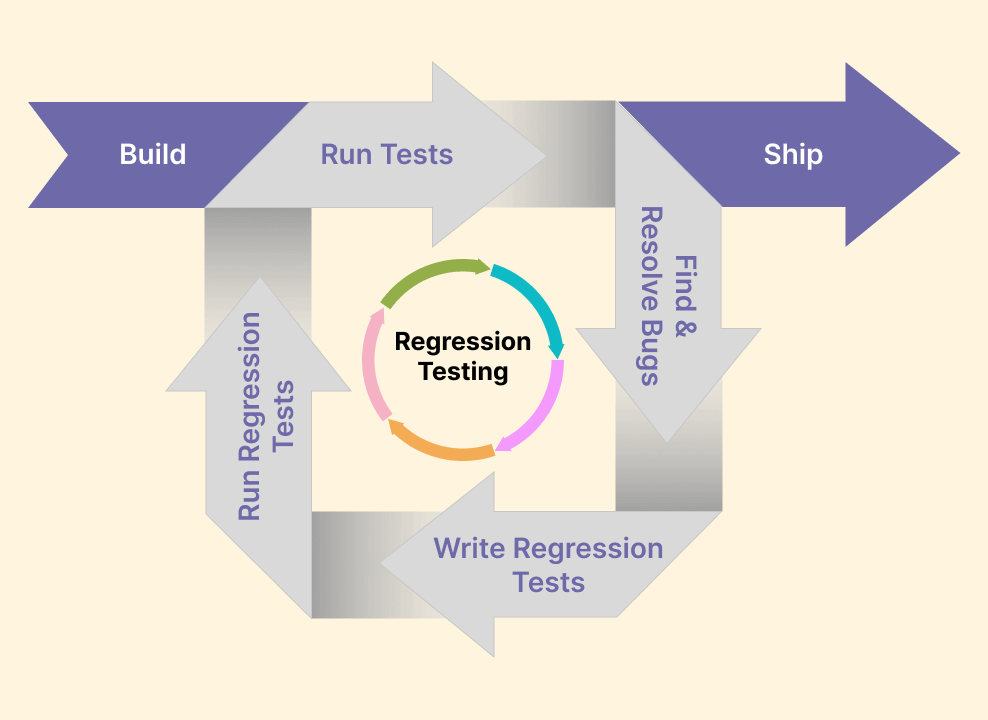
This ensures the operational working of the processes and procedures are aligned in the system. It determines the software's recovery, reliability, maintainability, and compatibility before its release.

**Regulation Acceptance Testing**

It is also known as Compliance Acceptance Testing. It considers whether the developed software conforms with the regulations. In other words, it mainly tests the software against the defined rules and norms the government sets before the release.

**Regression Testing**

Regression testing is considered part of functional testing as it validates the intended function of the software. Due to the new changes required in software development, developers often make some enhancements or code fixes. However, such changes may impact the other functionality of the application. Regression testing ensures that new changes do not hamper or affect the existing features or give rise to a new bug in the application.

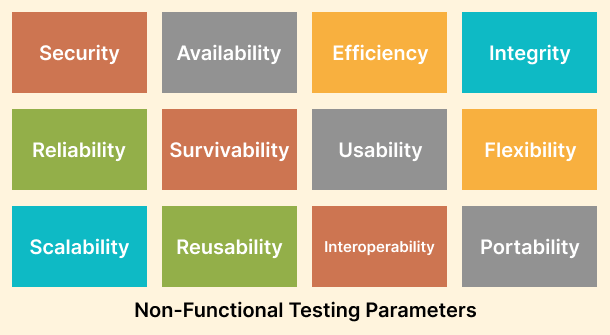


It can be performed using some automation testing tools like Watir and Selenium. It includes the re-execution of a suite of test cases that were used in previously passed tests.

**Non-Functional Testing**

Non-functional testing is a type of testing that considers the non-functional aspect of the software like performance, usability, reliability, portability, efficiency, security, and others. It takes into account the behavior of the system and the end-user experience.

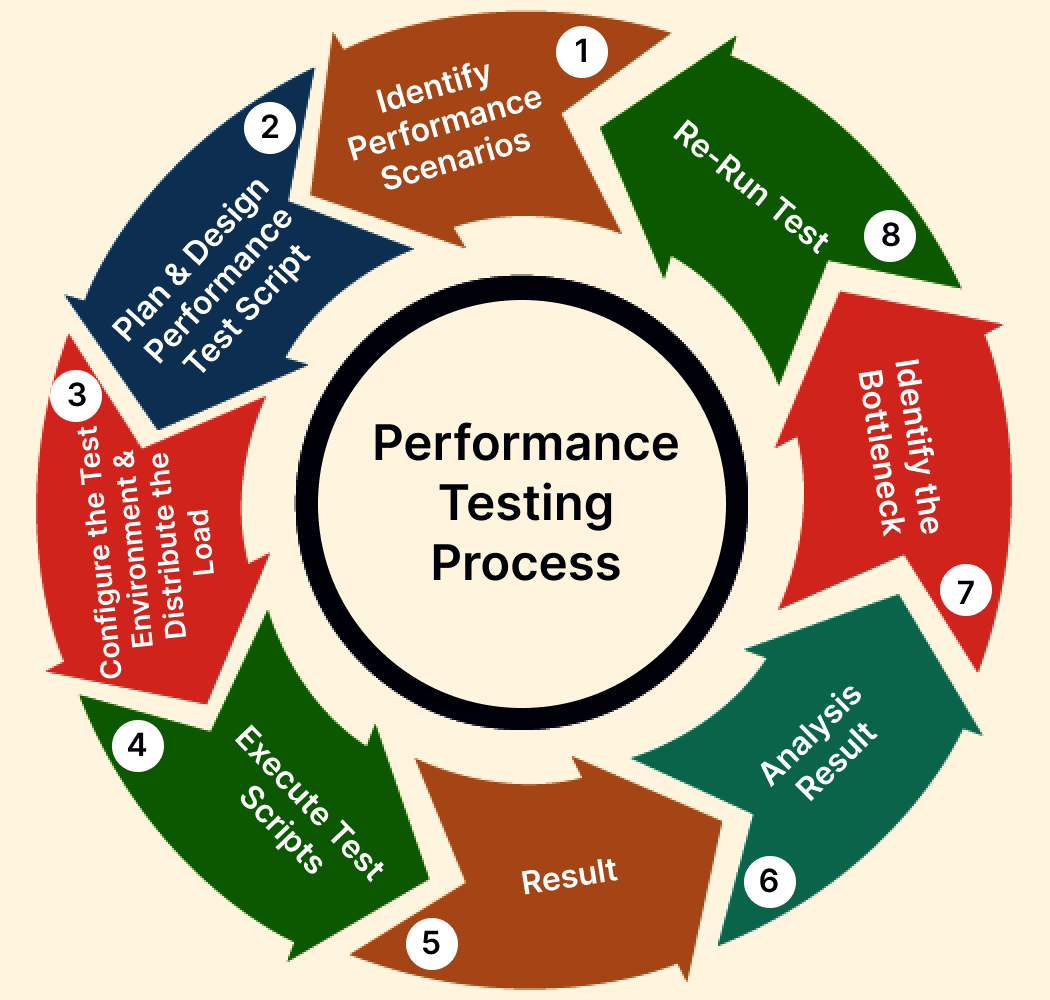
Such testing is mainly measurable and helps lower the production risk and monetary value. The parameters of non-functional testing are presented in the below illustration.



Non-functional testing holds different methodologies to perform testing. Some of them are as follows:

**Performance Testing**

It ensures the performance goals of the software application, like response time and throughput. With performance testing, the factors influencing software application performance, such as network latency, database transaction processing, data rendering, and load balancing between servers, are revealed in testing. It is mainly executed using tools like LoadRunner, JMeter, Loader, and others.



The types of software application testing under performance testing are highlighted below:

**Load Testing**

It ensures the stability of the software application with the application of the load, which can be equal to or less than the intended number of their users.

For example, your software application manages 250 users simultaneously with a noted response time of three seconds. In such a situation, load testing is conducted by applying a load of up to 250 or less 250. The main aim is to validate three seconds of response time.

**Stress Testing**

It ensures the application's stability and response time by giving a load of a number above the intended number of users of the application.

For example, an application is designed to manage 5000 users simultaneously with a response time of five seconds. Stress testing is performed by applying a load of 5000 and above users. Followed by noticing the response time.

**Scalability Testing**

It checks the application's scalability by applying a load of more than the designed number and investigates the point where the application can crash. It is done to confirm the ability of software to be scaled up and down to align the changes done.

For example, if an application handles 3000 users at a time and the response time is five seconds. Then, scalability testing is conducted with a load of more than 3000 users. Gradually, the load value increases to figure out the exact point at which the application can crash.

**Flood Testing**

A large volume or set of data is transferred to the database to check the stability and response time of the testing. In other words, the QA team uses flood testing to investigate the capacity of the database to manage the data.

**Endurance Testing**

It is executed to ensure that a software application or the system can handle the continuous load over the long term. In other words, it ensures that the application works properly for a more extended period.

**Usability Testing**

In simple terms, the tester tests the application's working in terms of user-friendliness. It is also categorized under black box testing, and application tests ensure easy utilization of the user interface by the user. Testing is done in three aspects: convenience, usability, and learning ability.

Usability testing is done to ensure the quality and easiness of application usage. It can be explained with an example. A gaming application’s usability testing checks whether it is operated by both hands, the color of the background, the vertical scroll, and others. The type of usability testing includes the following:

* **Cross-Browser Testing:** It involves testing the application on various browsers, operating systems, and mobile devices. Cross-browser testing ensures the application's compatibility, as it should work on all different browsers, mobile devices, and operating systems.
* **Accessibility Testing:**This type of testing help defines the accessibility of the software for people with impairments. For example, in accessibility testing, various checks are done, like font size and color for visually impaired people and color blindness.
* **Exploratory Testing:** It is considered informal testing to identify the application and address the existing defect. Based on business domain knowledge, exploratory testing validates the application.

**Security Testing**

Software testing involves tests for security that unmask the software application's risks, threats, and vulnerabilities. It intends to prevent malicious attacks and identify loopholes in the software system. It involves two crucial aspects of testing-authentication and authorization. Security testing is an important aspect, as it makes the application secure and able to store confidential information when required.

It also checks the behavior of the software related to attacks from hackers and how it should be maintained for data security upon noticing such attacks.

Security testing is of different types, including the following:

* **Penetration Testing:** Pen testing evaluates the system to validate for vulnerabilities to external hacking attempts. This test is mainly executed by the authorized cyberattack on the system to figure out the limitation of the system with respect to security. The operations like SQL injection, Privilege Elevation, session expiry, and URL manipulation are performed to execute security testing.
* **Vulnerability Scanning:** It is executed using automated software to scan or screen the system on the ground of vulnerability signatures.
* **Security Auditing:** It includes the internal inspection of applications and OS for security limitations. Further, an inspection of code line wise is also conducted to perform the audit.
* **Security Scanning:** It is mainly done to find system and network weaknesses and then provide answers for lowering associated risks.
* **Ethical Hacking:**It includes hacking an organization's software system. Here, the primary intention is to uncover the security flaws in the system.
* **Portability Testing:**This type of testing is performed to test how changes in the environment can lead to modification in the performance of the software. For example, it checks how the software operates in different versions of OS or web browsers.

Basically, such testing is done if the customer proposes utilizing software applications on multiple platforms. However, it can also be regarded as the subset of system testing and is mainly executed after integration testing.

**Other Types of Software Testing**

Software testing involves various approaches to ensure the application's quality, performance, security, and functionality. Some other types of testing known in the successful development of software applications are briefly highlighted below:

**Graphical User Interface (GUI) Testing**

GUI testing is performed to test whether the graphical user interface of the software application is working appropriately as per the requirement. It checks the functionality as well as defines its adherence to quality standards.

Some of the common aspects tested in GUI testing are:

* Layout
* Labels
* Captions
* Icons
* Links
* Content
* Buttons
* Lists

**Non-Graphical User Interface Testing**

The testing of anything other than a graphical user interface comes under non-graphical user interface testing. For example, it tests the command line interfaces, batch process, and other events, instigating specific use cases in the application.

**Mutation Testing**

It is the category of white box testing and is executed by modifying the source code of an application and then validating whether existing test cases can recognize such defects in the system. Since the changes done are minimal, it does not impact the functionality of the applications.

**Risk-Based Testing**

The functionality is tested on the ground of the priority of the business and is more prone to failure. It is done by setting the priority for all functionality. Followed to this, the high-priority test cases are executed, proceeding with medium and low-priority functionality.

Noting the different types of testing, it is equally important to know about the software testing approaches. This is explained below in the given section.

**Approaches to Software Testing**

Software testing includes endless types, and each of those is performed using different approaches. Basically, the testing approach can be understood as the method or strategy utilized to execute the test systematically.

Manual and automation testing approaches are commonly used methods to execute software testing. Each approach has specific requirements and purposes at the diverse stage of the software development cycle. Let us look into this in more detail.

**Manual Software Testing**

Manual testing involves the execution of testing without the use of any automated tools. While testing, all the test cases are performed by the tester manually as per the end user’s view. Some crucial aspects of manual testing involve:

* Able to find hidden and visible bugs in the software.
* Perform manual testing before executing automated testing.
* Although it involves a considerable amount of time and effort but provides a bug-free application.
* Able to uncover human-related errors like user interface problems.
* Since it involves human input, it is prone to error.

Manual testing includes three major types, explained in the previous section:

1. White Box Testing
2. Black Box Testing
3. Gray Box Testing

**How to Perform It?**

All the different types of manual testing follow a set of steps. Below are the steps required while performing manual testing:

* Evaluate the needs of software project documentation and guides.
* Form a test plan.
* Covering all the test requirements specified in the document, test cases are written.
* The QA lead then reviews the test cases.
* On approval, test cases are executed, and bugs are detected.
* On finding bugs, report them, and then fix them by re-running the failed test cases.

In a nutshell, manual testing is impossible to avoid in software testing because it is a continuous process. It needs human verifications regularly. However, this demands an equal balance between automation testing. Even though the agile approach toward software development is inclined towards automation testing, the manual approach is required.

Now let us see what does automation testing do?

**Automated Software Testing**

Automation testing is an approach to automate web or mobile app tests using automated testing tools and scripts. In other words, automation tools automatically run tasks in a pre-defined pattern. It is preferred by the tester to run software testing due to the following reasons:

* Automated testing increases test coverage, effectiveness, and test speed.
* Able to identify human-related errors.
* Save time in testing.
* Gives more control over the testing process.

However, you might think, when should we do automated software testing? Let's see this with an example. In regression testing, we test the changed code and its impact on other software application functionality.

Using manual testing in this approach seems to waste time and effort. The main reason is the need to perform a complete application test again. Therefore, rather than expecting humans to repeat the same test with similar speed, energy, and speed, it is preferred and logical to use software tools to execute such tests. Here, automation testing comes in.

When starting with automation testing, you should consider that not every test can be automated.

Some examples of a test that uses the automated software testing approach include:

* Smoke testing
* Data-driven testing
* Performance testing
* Functional testing
* Regression testing
* Unit testing
* Security testing

While performing automated testing, a certain set of steps requires to be followed to get accurate and fast output.

Read the section below on how to execute automated software testing.

**How to Perform It?**

While moving from manual to automation testing, it is crucial to consider the realistic goal by setting smaller test cases, addressing aspects not requiring automation, and others. However, several crucial stages of automated testing include the following, which run parallel with the software development life cycle.

* **Defining the scope:** To overcome any future challenges, the goal and objective of automation testing need to be penned down. It should also include budget and resources.
* **Select the right automation testing tools:** To perform automated testing; tools are the prerequisite factor. However, choosing the right automation tools depends on the web application’s technology being tested. Some of the tools include Selenium, Cypress, Playwright, Appium, etc.
* **Creating a test script:** Write automated test scripts which mimic the activities of users and verify the behavior of the software.
* **Test environment set-up:** The test environment, including software, hardware, and data, requires to be set up for test execution.
* **Execute test:** Then run the automated test by monitoring the output.
* **Debug and maintenance:** Noticing bugs and errors during the test, debugging, and maintaining the automated test scripts.
* **Continuous integration:** Automation testing requires to be integrated into the software development process. The main reason is executing the test every time on changing the code.

While performing automated testing, selecting the right testing platform is crucial for more reliable and scalable test automation. It is understood as the infrastructure where all automated tests run. It includes different versions of devices, OS, and browsers on which software testing is executed. It also provides parallel testing, where multiple tests on multiple devices run simultaneously.

As we discuss the different approaches to testing, specific metrics are used to check and determine the testing quality, like defect metrics, defect severity, etc.

Let's get into this in detail.

**Software Testing Metrics**

The primary focus of testing is to deliver high-quality software applications. To accomplish this, software testing metrics are considered to measure and quantify the testing process. Some metrics against which testing is executed are explained below:

* **Test Coverage:** It takes into account the extent to which the application is tested and evaluated in terms of functionality, need, and code coverage.
* **Base Metrics:** QA team collects the test data during test cases' development and execution. The test reports are generated, and the metric is shared with test leads and managers. It considers the total number of test cases and test cases completed.
* **Calculated Metrics:** Calculated metrics are created from base metrics data. Such metrics are used by testers to track the development of software applications and their progress.
* **Defect Metrics:** These give information on the aspects of software quality, like functionality, usability, compatibility, and installation stability.
* **Defect Severity:** Help see how defects can affect the quality of the software application.
* **Test Case Efficiency:** It measures how effectively test cases detect bugs.
* **Defect Density:** It is the number of confirmed bugs in a software application during the development period, divided by the software application size.

Such metrics allow tracking and monitoring of the quality of the testing approach over time. Along with this, it helps identify areas of improvement and decide how to allocate resources, and prioritize efforts.

Let us look at the different strategies used for software testing.

**Strategies for Software Testing**

In software testing projects, two crucial factors that require consideration are strategy and investment. Here, strategy is a priority as it gives information on techniques and tools needed for testing. Some of those strategies which can be used are explained below:

**Static Testing Strategy**

The test where the developer reviews the code before pushing is a form of static test. Here, the system's quality is evaluated without running the system. It allows early bug detection and fixation.

**Structural Testing Strategy**

It is one of the strategies under unit testing, where the system is tested as a whole and validated on real devices to find all bugs. It may be called white box testing as it is executed on different components and interfaces to find defects in data flows.

**Behavioral Testing Strategy**

It addresses the system's working in terms of performance, configuration, workflow, etc. The main focus is to test web applications or websites as per the user’s view, hence called the black box test. It can be done through both manual and automated testing approaches.

However, having different testing strategies, you need to know the basis for selecting appropriate software testing approaches. You can consider the following factors:

* Associated risk during the testing.
* Requirement of the stakeholders.
* Regulation of the organization.

Additionally, software testing strategies mainly focus on the recognition of bugs. The best approach to detecting all bugs is to run the application on real devices and browsers. However, manual and automated testing should be the central area for testing a website or web application. Automated testing should complement manual testing to detect all the bugs.

**Software Testing Tools**

Software testing is made easier with the availability of testing tools. They support various test activities from planning, gathering, building creation, test execution, and analysis. For the automated testing approach, different ranges of tools are available, which can be used as per the needs. Let's get into details about such tools categorized under automation testing tools.

Some of the most trending automation testing tools for software testing are explained below:

**Selenium**

Selenium is an open-source automated testing tool that tests web applications across different browsers and operating systems. It is one of the leading automation testing frameworks for your web testing requirements. If you are testing an application in a browser and want to expedite the process, you can automate the testing process with Selenium.

**Cypress**

Cypress is one of the popular front-end automation testing tools for the modern web. It is based on JavaScript and uses DOM manipulation techniques directly in the browsers. You can write unit tests, end-to-end tests, and integration tests with Cypress. It does not necessitate adding explicit or implicit wait commands.

**Playwright**

Playwright is an automation testing framework based on the Node.js library and another chosen favorite for a larger audience. It automates Chromium, WebKit, and Firefox using a single API. Playwright supports multiple programming languages like C#, Java, Python, and Node.js.

**Puppeteer**

Puppeteer is a Node.js library and allows headless browser testing for Google Chrome. It uses commands in JavaScript that allows performing test actions on Chrome browsers. Puppeteer offers high-level API in situations where control of Chromium or Headless Chrome is needed with DevTools Protocols.

**Appium**

Appium is an open-source tool for mobile automation testing for Android and iOS apps. It also works for hybrid apps. Appium can be used for automated functional testing that improves the overall functionality of mobile applications.

Overall, there are an endless number of tools available for easy automation. You must only cross-check your business requirements to choose the appropriate testing tools.

**Where to Test Software Applications?**

Testing ecosystems contain various techniques that testers can choose from depending on their requirements. Testing is intended to ensure that the application under test is reliable, secure, and safe.

There are two methods to perform software application testing:

1. On-premise testing
2. Cloud testing

**On-Premise Testing**

On-premise testing involves testing software applications on local machines, systems, or devices in an office. As a result, it entails a great deal of responsibility. Maintenance, monitoring, and upgrading the machines and software, and installing and upgrading - you'll need everyone on board. Furthermore, on-premise testing is quite expensive and time-consuming.

**Cloud Testing**

Cloud testing evaluates web applications (or websites) for scalability, performance, security, and reliability. This testing is performed on a cloud computing environment hosted by a third party, which houses the necessary infrastructure.

There is a lot of talk about transforming businesses digitally. For those who want to embrace digital transformation, it's recommended to choose cloud-based testing over on-premise testing. There are numerous benefits of cloud testing. It is here to stay so you can keep ahead of the curve with the least operational overhead.

**How to Perform Software Testing on the Cloud?**

Cloud testing is effective for web and mobile application testing without worrying about the local test infrastructure. Testing on the cloud helps you leverage different browsers, devices, and various operating systems and eliminate coverage limitations for OS, devices, and browsers.

You can perform manual and automated web and mobile application testing with cloud-based platforms. It allows you to perform real-time testing, real-device testing, and automation testing on various browsers, browser versions, devices, and operating systems.

**Challenges and Solutions in Software Testing**

At present, software application testing is reinforced with regular updates to meet the changes in user expectations and technology. The rise of the Agile approach made it easy for testers to update the software at lightning speed. However, it brings a few challenges to software testing.

Let’s shed some light on those challenges and their solutions.

**Challenge 1: Inadequate Communication**

Poor communication within the team regarding the development of correct test cases can be one of the challenges. Considering that the right test cases cannot be prepared unless they know the business and technical needs is crucial.

**Solution:** Always collaborate with the testing and development teams while doing software testing.

**Challenge 2: Difference in Testing Environment**

Development of software requires working effortlessly across diverse device and browser platform combinations. However, the availability of thousands of mobile devices on the market created a significant challenge in software application testing. Although emulators and simulators are one option for overcoming this, they do not confirm the application's primary functionality in real-world user scenarios.

**Solution:** A cloud-based testing platform, such as LambdaTest, provides real-device cloud access, allowing users to test across 3000+ real browsers and devices. They are also integrated with automation test tools and frameworks that ease software testing.

In addition to the above challenges, software testing is often misunderstood as a debugging method by some new testers who often mistake software application testing as a debugging method. However, both are very different in their meaning.

Learn in detail about the major difference between software testing and software debugging to have a clear understanding.

**Software Testing vs. Software Debugging**

Novice testers often get confused with the terms "Software Testing" and "Software Debugging." They may sound similar, but they are different. Some of the key differences could be understood from the below-given points.

| **Software Testing** | **Software Debugging** |
| --- | --- |
| It should be done throughout the Software Development Life Cycle. | Software debugging is done when software application testing gets over. |
| It unmasks the defects. | Debugs remove the identified defects by locating them. |
| It is part of the software development cycle. | Debugging is part of the testing process. |
| It initiates as soon as the development of software begins. | Debugging initiates when the testers report any defects. |
| Verification and validation of software are involved in testing. | In debugging, the real cause behind the bugs is rectified. |

**Best Practices for Software Testing**

To ensure the perfect execution of software testing and deliver a high-quality product, it is vital to incorporate its best practices. The following are the best practices:

* Before testing begins, planning and preparing the test by defining the scope, identifying test objectives, and selecting appropriate tools is crucial.
* Test cases should be defined clearly and comprehensively based on all aspects of the software. Further, it is crucial to note that test cases should be associated with test objectives.
* Make use of automation testing tools to run repetitive and time-consuming tests.
* Initiate the testing process early in the software development process. Continued execution of the testing process throughout the entire development lifecycle is recommended.
* Perform testing on multiple platforms and devices to confirm that it works with all test environments.
* Always have thorough tests using different testing techniques like unit testing, functional testing, integration testing, and performance testing.
* Work in close association with the development team. It helps ensure testing integration into the development process and the timely fixation of bugs.

**Conclusion**

The detailed discussion on software testing would help gain a good understanding of its concepts. It has been an essential part of the software development process. It ensures the quality, performance, and reliability of software.

It can be said that with a continuous process of analyzing and improving testing, the organization provides its updated practice of software testing in line with the latest industry standards and best practices.

Finally, combining manual and automated testing methods and practices such as early and frequent testing and team collaboration allows for the detection and timely correction of bugs.