Types of Software Testing

❖ Topics Covered

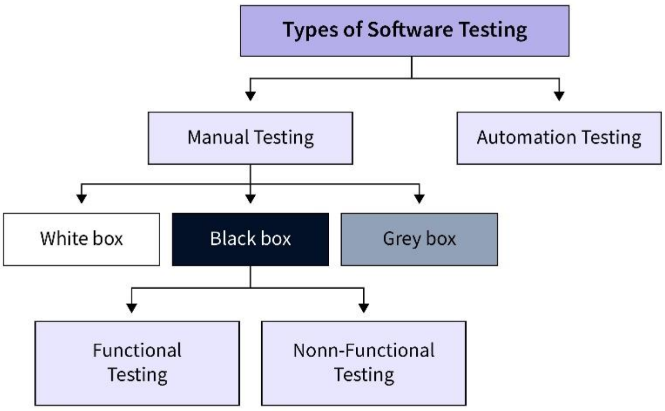
- Principle of Software Testing
- Manual Testing
- Automation Testing
- Advantages of Software Testing
- Disadvantages of Software Testing
- Additional Resources
- Conclusion

Various software applications are developed to facilitate the standard exercises of human-like online shopping, consultations, automated traffic control systems, automated machine arms in factories, etc. Software applications should perform smoothly and accurately, while failure can be disastrous for organizations and people. Thus, making software testing an integral part of the software development life cycle.

❖ Principle of Software Testing

- All the tests should meet the customer's requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned before implementing it
- It follows the Pareto rule (80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

- There are two types of software testing broadly :-
  - Manual Testing
  - Automation Testing



❖ Manual Testing

It is a way of physically testing the software without utilizing automated tools like coded test scripts. Exclusive test cases (i.e., a set of conditions to test the application's functionality and verify expected results) are written and implemented to test their pass or fail status. It guarantees whether the application is working, as referenced in the product requirement document or not. **Manual testing** is compulsory for each recently developed application before automated testing. This testing requires extraordinary endeavors and time, however, it guarantees bug-free software or applications.

According to testing criteria, manual testing can be divided into:-

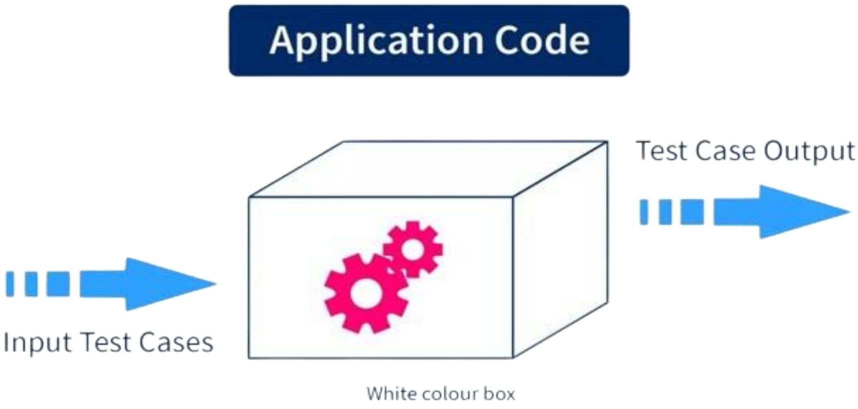
1. White Box Testing
2. Black Box Testing
3. Grey Box Testing

1. White Box Testing :

The primary aim here is to analyze the internal code structure, data types used, internal code design, the flow of inputs, and respective outputs from the application.

The term “white-box” is used to focus on the inner perspective of the application. The transparent or white box signifies the ability to see through the product’s external shell into its internal work.

The developer does white box testing, where he goes through every line of the code. The developer finds and fixes the bugs.

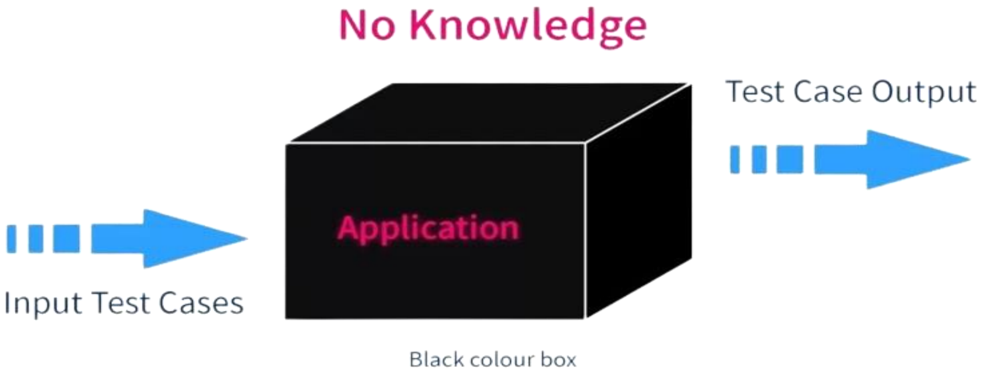


The diagram shows that the internal code structure is known to the developer. Test cases are passed to check if desired outputs are obtained.

2. Black Box Testing :

When the functionalities of the software application are tested without knowing the internal code structure, it is called Black Box testing, while in white box testing, the internal code is known to the tester. The term “Black box” is used as the tester is unaware of the internal code structure of the application.

Example:- Black box testing is performed when website functionalities like load time, links present on the website, and server response time are checked without having access to the HTML code.



Black box testing is performed by a test engineer or sometimes a developer in some organizations, who creates test cases to check the efficiency and accuracy of the application. All test cases are planned considering the input and desired outputs. The tester is aware of a particular input's output but unaware of how the internal code is getting executed.

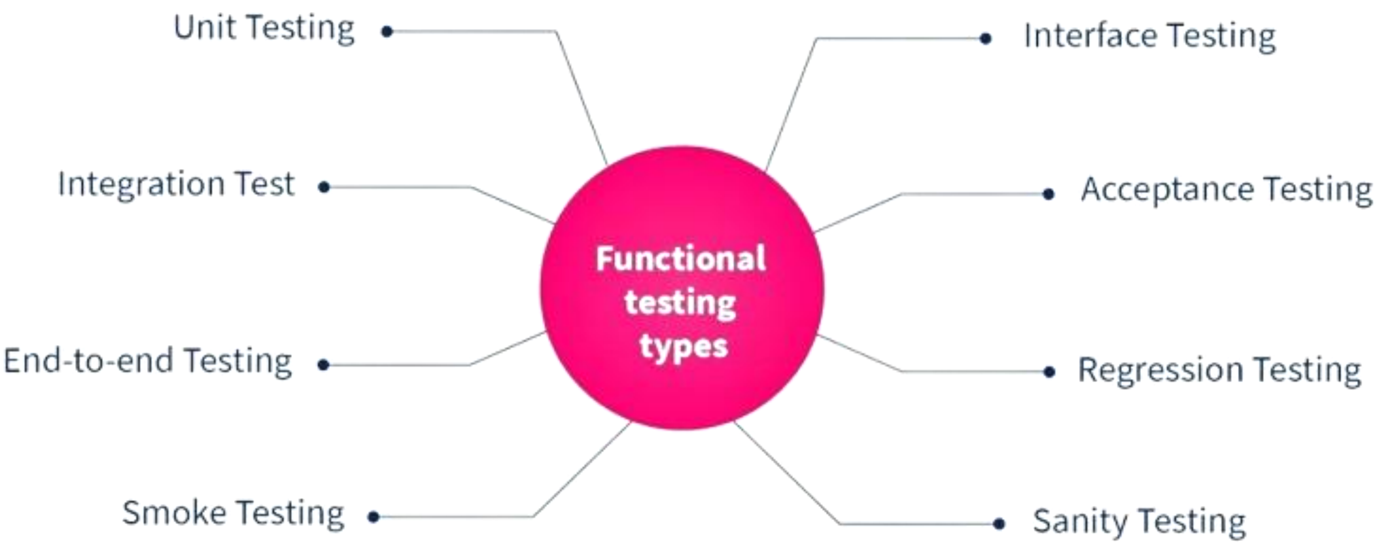
Black box testing can further be divided into:

a) Functional Testing
b) Non-Functional Testing

a) Functional Testing

Functional testing is a software testing method that checks if the core purpose of the software is fulfilled i.e. the functional necessities are ensured. The focus is to check the software's functionality by providing input and checking the desired output against functional necessities.
Functional testing objectives include testing main functions, ease of usability, and extent of accessibility of the system. The testing can be done either manually or using automated tools. Some of the popular tools are JUnit(mainly used with Java applications), Selenium (an open-source tool for both web and desktop applications), soapUI(mainly used for web application testing), etc.

Functional testing can be divided into multiple parts:



i) Unit Testing

It is the primary level of functional testing. It is normally focused on the smallest unit of the code structure. Testing is done by the developer or programmer on an individual unit (i.e. a module/component of code that is independent of the rest of the code) or a group of related units to check if they produce the desired output.
But in unit testing, assessment of all execution ways is impossible, so it alone cannot get each bug in the program.

Example:- If a developer can log in using the correct username and password and gets redirected to the welcome page, it successfully marks testing of the login feature (so, a unit).

ii) Integration Test

As the name suggests, integration testing examines units or individual parts of the product in a group. It is vital to check the collaboration between incorporated units(i.e. individual modules/components of code).
Generally, integration testing is carried out after unit testing to guarantee every one of the units works in agreement with the other. It is also done when support libraries are utilized alongside the code.
As integration testing covers several modules simultaneously, identifying the error's source becomes increasingly difficult.

Example:- Integration testing can check if the signup and login feature work hand in hand i.e. the application should store details of a new user at the time of sign up and those details can be used when the user logs in again. Integration testing ensures that login details also work with other features like editing profiles, adding new information, etc.

Some tools for unit and integration testing include - Jasmine, Mocha, Citrus, FitNesse, etc.

iii) End-to-End Testing

End-to-end testing includes testing an application's execution from start to end. This testing essentially plans to duplicate genuine user scenarios so the application can be approved for information and data integrity. It verifies data flow within the application and is generally conducted on finished products or applications.

Basically, end-to-end testing helps test the interaction of each application activity with databases, network connectivity, dependencies, hardware, and other applications.

Some tools that help in end-to-end testing include:- Cucumber, Jasmine, Protractor, etc.

iv) Smoke Testing

Smoke testing aims to test an application's essential and basic elements before doing rounds of thorough, rigorous testing. Here, we focus on the positive workflow of the software and check the output for valid inputs, not invalid ones.
Smoke testing is carried out at the time of handing the software over to the testing team from the development team.

v) Sanity Testing

Sanity testing is carried out to ensure that every one of the bugs has been fixed after the build and that no additional issues arise on account of these changes. It also guarantees that the code alteration does not affect the associated modules.
For sanity testing, documentation is not compulsory. That’s why it tends to be executed in less time. It is done by a quality assurance engineer who ensures the application is working fine after deploying a new feature.

vi) Regression Testing

Regression tests are performed to guarantee and check the previously developed, and the tested application performs well even after introducing new changes or adding new features in the software or application. It checks for any new error in the current application and is a verification method for the application.
There is a need to re-run old tests to guarantee that new functionalities do not introduce old defects or make new ones through software regression testing.

It sounds similar to a smoke test, doesn’t it? Let’s see what’s the difference:
Smoke testing is utilized to check whether the application build is steady as the first thing, and it can be subjected to further testing by the testing team, while the regression technique is a maintenance test to ensure code changes do not affect previous functionalities of the application.
Regression testing is also used when new features are added to existing software or application.

vii) Acceptance Testing

User Acceptance Test is a sort of testing done by some of the customers or domain experts who are known to the application before the product's final release.
There can be some minor bugs that can be recognized only when the product is introduced to the end-user to test the working in real-time scenarios, thus, acceptance testing is vital. This results in increased satisfaction among the clients, who can test the product for themselves.

viii) Interface Testing

Interface testing is done to verify whether the interaction between two software systems is done effectively. The interface represents a connection between two components, such as APIs, web services, etc. An interface comprises commands and messages to facilitate communication between programs.

Interface testing mainly comprises two segments:

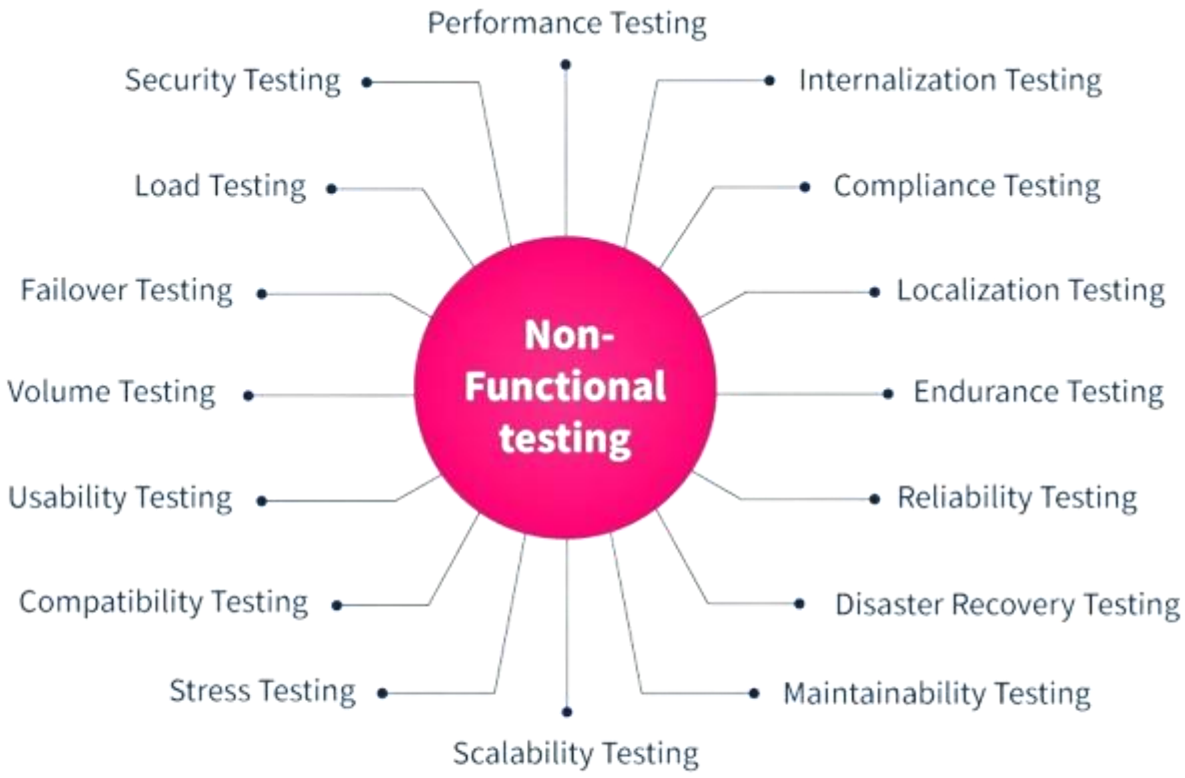
- Web server and application server interface It checks the connectivity between the server where your website is hosted and the server on which you created your application.
- Application server and Database server interface It checks connectivity between the server where your application is created and the database server where one or more databases are hosted

b) Non-Functional Testing

Non-functional testing tests how the software or application behaves under different circumstances by examining dependability, load test, execution time, and application performance.

It streamlines how the application is installed, set up, and managed. It helps to ensure that your application can handle the load and remains stable under higher user traffic. Some test cases include: Load time for the application should be less than 5 seconds, all web images should have alt tags, application recovery after a crash, application security, software should be installable on all versions of Windows, and Mac, etc.

Non-functional testing can be divided into:



i) Performance Testing

Performance testing assesses how an application reacts under a particular workload in terms of stability and responsiveness. Performance tests are regularly performed to check speed, reliability, and application size. The performance indicators include:

- Response time for browser, network response, page load
- Server request processing time
- Acceptable simultaneous client volumes
- Processor memory utilization

Example: verify if the response time is not more than 4 sec when more than 1000 users access the application simultaneously or check the database execution time when 500 or more records are inserted/ accessed simultaneously.

ii) Security Testing

The target of security testing is to track down every one of the expected ambiguities and vulnerabilities of the product to protect it from attacks to breach the security system of the product. It is a necessary part of software testing as it assists us in recognizing all possible risks and helps developers fix those.

Attacks can be of any type i.e. client-side attacks, authentication, authorization, information disclosure, etc.

Some tools for security testing include SonarQube, ZAP, Netsparker, etc.

iii) Load Testing

Testing the performance of an application by applying some load i.e., either less or equal to the ideal load, is known as load testing. It produces real-time user scenarios.

Here, load implies that N-number of clients utilizing the application or sending the server requests simultaneously to the server. Load testing will assist with recognizing the maximum working limit. It is primarily used to check the Client/Server’s performance.

Some load testing tools include LoadNinja, Apache JMeter, NeoLoad, etc.

iv) Failover Testing

Failover testing approves an application’s capacity to be able to allocate additional resources and shift tasks to backup systems. This decides whether a product is fit for dealing with additional assets like additional CPU or servers during a system failure or at the point the product arrives at a performance limit.

Failover testing is essential for governmental applications, financial applications, trading platforms, and telecom platforms.

v) Compatibility Testing

When the product is stable, we move it to the creation phase, it will be used by a range of customers using different platforms, and they may confront some compatibility issues, to avoid these issues we perform the compatibility test.

Generally, compatibility issues consist of user interface issues like arrangement, cross-over, and dispersed issues. The domains tested in compatibility testing are software(Operating systems), hardware(processor, RAM, etc.), network parameters(bandwidth, operating speed), and mobile platforms(Android, iOS).

vi) Usability Testing

It checks the ease of using the product in a user-centric environment. It requires extensive knowledge of the product to do so. Generally, usability testing is performed from an end-user perspective to confirm whether the product is proficiently working.

It assists us with fixing a few convenience issues in a particular site or application, in any event, ensuring and improving its excellence and functionality. The testing is done by real-life users, not by the development team.

vii) Scalability Testing

Scalability testing checks a product’s performance by expanding or reducing the load at specific scales. It is executed at a hardware, programming, or database level. It is the limit of a network, product, application, or process to code effectively when alterations are made in the product’s size or volume to meet an expanding need.

It also checks the ability of the application, the database system, and processes to meet an expanding need(scalability).

viii) Volume Testing

In volume testing, we will focus on the information flow rates rather than the number of users. It is likewise called Flood testing.

It is executed to break down the impact on the framework’s reaction time and conduct when the volume of information increases in the data set. In this testing, a large volume of data is followed up on by the product. Confirming that the framework isn’t in danger of overflow or information security issues is needed.

ix) Stress Testing

The stress testing aims to ensure that the product doesn’t crash if it lacks computational assets like disk space, memory, and service requests. It focuses on the error handling capacity of the application under heavy load or stress conditions rather than on regular behavior under the expected load.

Stress testing is utilized to check whether the application saved the information before the crash. It tests if the application can handle the load properly and also ensures to show an appropriate blunder message when the application is under pressure.

x) Maintainability Testing

Maintainability testing determines how easy it is to manage and maintain the application after it is deployed or when in use. It implies how easily we can analyze, detect, and fix bugs and test the product or application. Testers should add reviews and analysis at the time of dynamic testing while coding walkthrough and inspection.

xi) Compliance Testing

Compliance testing is done on the product to check if it meets a defined set of standards before it’s released into production. A leading group of regulators and compliance experts ensure that the compliances are met or not. This board checks whether development teams are fulfilling the standards or not.

A compliance test is also known as a conformance test. The standards are laid by organizations like IEEE (International Institute of electrical and Electronics Engineers) or W3C (World Wide Web Consortium), etc. It can also be done by an autonomous/third-party organization specializing in compliance testing.

xii) Reliability Testing

Reliability Testing is implemented to ensure the product or application works reliably in each possible technological environment for a specific period. It checks whether the software can achieve an error-free activity in any environment or not.

It assists with ensuring we deliver a quality end product.

xiii) Endurance Testing

Endurance Testing tests the product's performance under specific load conditions over a broad period. Endurance testing is useful to check if there are any memory leaks in the system and response time over an extensive period. It must be ensured that response times after longer use don’t degrade over time and under heavy load. Here, testing is performed for long hours, such as 15 hours, or 90 hours, as compared to other methods of testing like load or stress testing.

xiv) Disaster Recovery Testing

Various functionalities are interrupted during a disaster or system failure, like information handling, server operations, visualization, and data center operations.

Disaster Recovery Testing guarantees that the product can recover information and critical applications and reestablish processes and connections after critical failure or interruption in product services.

xv) Localization Testing

Localization testing is done to check the performance of a product in correspondence to a specific culture, locale, or region. It improves the product’s linguistic and cultural aspects for a specific region. It customizes software as per specified region or culture.

The product’s user interface, default language, currency, date, time arrangement, and documentation are planned according to the designated nation or region, or culture.

xvi) Internalization Testing

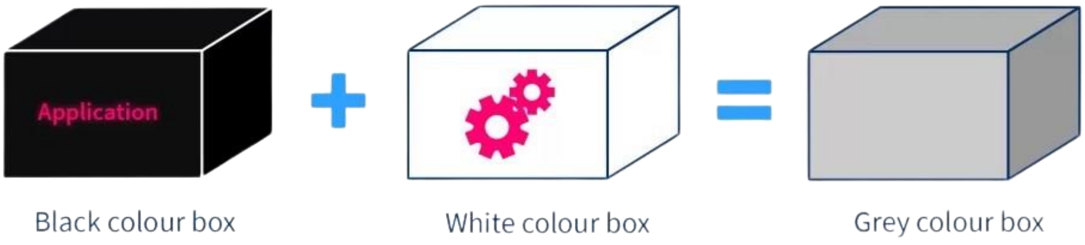
Internalization testing ensures that the behavior of the product is consistent across multiple regions or cultures. The main aim is to check if the program can deal with all worldwide support without failing which may cause information loss or integrity issues. It checks that the product is appropriate for any regional setting.

3. Grey Box Testing

It is a blend of black box and white box testing since it includes access to internal code structure to configure test cases as white box testing and the functionality of the software is tested as black-box testing. The test cases are designed based on algorithms used, security, database, and required description of the behavior of the program.

Grey box testing is named so because it is a mixture of white box and black box techniques and provides a semi-transparent view of code structure and functionality. The overall quality of the product is improved using a single technique.

Example: While testing a website, if any error comes up while opening a link in the website, changes can be made to the HTML code. Here, the tester does black box testing by examining the website and simultaneously does white box testing by changing internal code to fix the error.

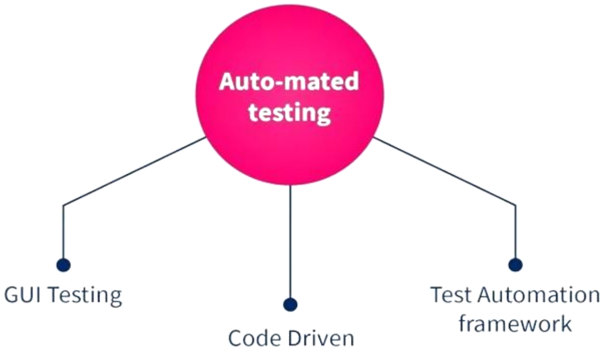


❖ Automation Testing

In automation testing, test scripts are executed rather than running tests and searching for bugs on your own. for the bugs. With the assistance of automated testing techniques, we can handle the test execution and analyze the derived output and expected result.

In automated testing, the test engineer will compose a test script rather than manually developing test cases as in manual testing techniques. As a result, the need to develop test cases repeatedly is struck out. Automated testing techniques are way faster than executing manually written test cases. To perform automated testing, we need a test engineer who can compose the required test scripts rather than having a group of people who are over and again executing dreary manual tests.

Automated testing consists of the following parts:



1. GUI testing

Users interact with the application or software through a user interface. It can include display screens, keyboards, or a mouse.

The purpose of GUI(i.e. Graphical User Interface) testing is to guarantee the ease of functionalities of applications by checking the smooth operation of screens and controls like buttons, menus, images, links, and so on.

2. Code driven

In this strategy, the engineer will fundamentally focus on the execution of the test cases to analyze whether all the functions of the code are producing desired output or not.

3. Test Automation Framework

Using a framework for automated testing, will speed up the team’s testing and efficiency, further develop test accuracy, and will decrease the maintenance cost for tests. A framework sets guidelines that incorporate coding standards, test-data handling processes, or data on how to access resources.

❖ Advantages of Software Testing

- **Testing saves money**
  Notably, the longer the bug goes undetected in your product, the more troublesome and costly it is to fix it. It also costs you, clients, and customers. By implementing software testing all through the development phase of the product, you can save money and time after the product is deployed.
- **Prevents Emergencies**
  Errors in corporate software can prompt system failures, missing information, and server breakdowns. The possibility of such cases can be ruled out by implementing specific tests to ensure proper backup channels and secure data during such failures.
- **Secures the product**
  Nowadays, there are many situations where customers' personal data is leaked and used to their advantage. This is a grave threat to any individual.

As a particular application goes through testing, the client can be guaranteed that they will get a reliable item. The personal data of the client can be protected. Clients can get safe and protected items from malicious attacks by outsiders with the help of software testing.

- **Ensures quality**
  Software testing ensures that your product or application's quality is high, which is significant for the client. With higher quality, the customers will pay more. Considerably more significant is that with

selling top-notch items, you build a solid reputation and brand picture, things that are significant in the long haul.

- **Easy to make changes**
  As a new developer, changing parts of the older code structure can be difficult, but with suitable tests, you'll essentially know if you've led to failure in the functionality. As the addition part becomes simpler, the product remains ahead and more advanced.

- **Increased coordination and efficiency**
  Nobody wants chaos of defective applications and hurried fixes of bugs. Coordinated with testing at each step from the start of the development phase of your application brings in more organization and productivity with time. The testing and development teams work in parallel to increase the efficiency of the application.

❖ Disadvantages of Software Testing

- **Time-Consuming**
  Comprehensive testing can be time-consuming, especially for large and complex software systems. Testing each aspect thoroughly may extend the overall development timeline.

- **Costly**
  The resources required for testing, including personnel, tools, and infrastructure, can contribute to the overall project cost. Additionally, fixing defects found during testing may incur additional expenses.

- **Infeasibility of Exhaustive Testing**
  It's practically impossible to test all possible input combinations and scenarios for complex software systems. Therefore, exhaustive testing to guarantee complete coverage is often infeasible.

- **Dependence on Testers' Skills**
  The effectiveness of testing depends on the skills and experience of the testing team. Inexperienced or inadequately trained testers may overlook critical issues, leading to a false sense of security.

- **Limited Scope of Testing**
  Some aspects of software, such as performance under extreme conditions or security vulnerabilities, may not be fully addressed in regular testing. Specialized testing efforts may be required to address these aspects.

❖ Conclusion

- Software testing is gainful as it ensures higher returns on investments.
- Testing will guarantee better expectations in development and products, ensuring customer satisfaction and profits.
- The top organizations blend different testing approaches at different stages of their development cycle to achieve the best results.
- Several companies also provide testing services to test your software or application like Testlio, QAlified, Capgemini, QualiTest, etc.
- Remember that you don't have to play out all these tests referenced in this article for your product. What sorts of tests you should run depends on the product you're building.