Why Software Testing is Important

Software testing is a method used to determine if the current software product satisfies the expected requirements and ensure bug-free software. It involves running the various components of the software through their paces so that they can be evaluated with the use of manual or automated tools.

Software testing aims to discover gaps, errors, or missing expected requirements.

Errors may pop up in any phase of the software development life cycle (SDLC), and a few of them have been known to pass through undetected. That's why quality assurance is so important.

There's a high possibility that there may be design and functionality errors in the final code. Therefore, to release software with confidence in its quality, you need

to ensure that those errors and any other bugs are first caught and fixed before delivery. This is where software testing comes in.

A properly tested software product ensures security, dependability, and high performance, which translates to cost effectiveness, time efficiency, and customer satisfaction.

Testing is an integral part of the software development life cycle. Many companies trust their software developers to test their products, failing to recognize that in order for testing to be effective, it needs to be conducted by a qualified and experienced third party. Third-party testing services, although costly, is less than the cost associated with sub-par testing services.

Below, we discuss the leading reasons software testing is so important.

You will learn:

Reasons why software testing is important

1. It helps save money

   The resulting cost-effectiveness of the entire project is one of the top reasons for employing software testing services. There's the old saying that "prevention is better than cure" and this definitely proves true in terms of software bugs and security issues. The cost of fixing an issue increases exponentially as the software moves forward in the SDLC.

2. It improves security

   It is imperative to ensure that the personal details of users are protected from unauthorized access and use. The system also needs protection against vulnerabilities like ransomware and malware attacks. Traditional SDLCs carried out security testing at the end, however, conducting security assessments throughout the SDLC ensures more reliable software.

3. It improves product quality

   Properly tested software ensures product security, usability, and functionality, and performs as close to perfect as possible.

4. It achieves customer satisfaction

   If you release a faulty product, your brand reputation will be tarnished by bad reviews from dissatisfied customers. These negative reviews will ward off potential customers, which will result in revenue loss. Current customers with bad user experiences will also likely leave you for a

competing product. Their trust in your product will be lost and so will your sales from them. Software testing ensures that you find product faults before releasing them into the market.

5. It enhances the development process

When you continuously test software you are able to identify errors, bugs, and defects in the initial stages of the development process and fix them early enough to save the time it would have taken to fix them later in the process. A testing team working with the development team accelerates the software development process.

6. It makes adding new features easy

The older interconnected codes make it extremely difficult to add new features. However, adding tests to the mix allows you to know where any breaks in the code may have occurred so that you can fix them. Because software testing makes it so much simpler to add new features, it helps your software stand ahead in the market and beat the competition.

7. It determines software performance

Application or software with low or reduced performance negatively affects your brand reputation. If you release software without testing, poor software quality may be your result. Software testing helps determine the software's performance, and in effect achieves customer satisfaction.

What are different types of software testing?

The types of software testing generally fall into two groups:

1. Manual Testing

In this software testing process, test cases are executed without the use of automation tools, and from the point of view of the end-user. A human sits in front of a computer carefully carrying out the test steps to ensure that the software application works as expected, based on the requirement document.

The various stages of this testing process include unit testing, integration testing, system testing, and user acceptance testing.

Manual testing also includes exploratory testing, which is where testers explore the software to identify any errors in it.

2. Automation testing

Automation testing, which is also referred to as test automation, is where manual tests are automated to facilitate quicker constant re-runs of test scenarios. It is a software solution that utilizes special automated testing software tools.

With the exception of regression testing, test automation is also utilized in performance, load, and stress testing. Compared with manual tests, test automation saves time and money, improves accuracy, and increases test coverage. It must be noted, however, that automation does not aim to eliminate manual tests altogether, but rather to reduce the number of test cases to be run manually.

Types of automation testing include:

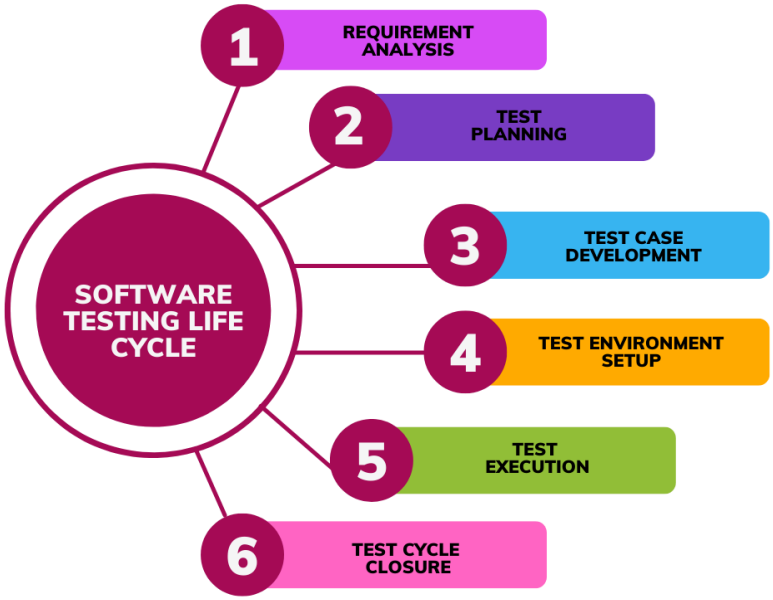
- **Functional Testing**
- **Unit Testing**
- **Integration Testing**
- **Smoke Testing**
- **Non-functional Testing**
- **Performance Testing**
- **Regression Testing**
- **Keyword-driven Testing**
- **Data-driven Testing**

Techniques in software testing

Black Box Testing and White Box Testing are the two major categories of software testing.

While Black Box Testing and White Box Testing have different functionalities, they both guarantee the best version of the software and that there are no glitches in the system.

What are the steps involved in the Software Testing Life Cycle?



The Software Testing Life Cycle (STLC) is one of the various approaches and strategies employed by testers in their effort to achieve optimum quality. It is a process where testers aim to satisfy the software quality requirements.

There are six steps involved in the STLC:

1. Requirement analysis

This is the first stage in the STLC. It is the process through which testable requirements are identified by testers. QA professionals interact with all stakeholders for an in-depth understanding of their requirements.

There are many times when requirements are unclear or missing, so without ideal requirements for the project, the software tester needs to utilize tools and methods to organize testing.

A feasibility study of automation is also performed at this point, if necessary.

Requirement analysis forms the basis for project plans and test plans.

2. Test planning

At this stage is where it becomes necessary to:

- **Devise a testing strategy;**
- **Identify risks;**
- **Choose the most suitable testing tools;**
- **Allocate roles in the team.**

All of these are recorded as part of the test strategy in the test plan.

3. Test case development

As soon as the test plan is fully developed, the QA team begins the development of test cases. This phase's principal objective is the preparation of test cases for an individual unit or feature of the project. These structural and functional test cases cover the areas mentioned in the test plan: points of verification, validation, and the product's functionality. These test cases help the team get acquainted with the program without having to read the entire code.

4. Test environment setup

This is an essential step in the STLC. It makes all the preparations required for an effective software testing process. This includes setting up a network, a test server, test devices and PCs, as well as the generation of test data.

The test team is not necessarily involved in this process, however, they should be prepared to conduct a readiness check to ensure that the configuration is testable

before being released for further testing. They need to also ensure the entry of the necessary test data into the system so that it is ready to be used.

This phase is sometimes completed together with the test case development phase.

5. Test execution

When the environment is set up, the test strategy is established, and the test plan is approved, it is now time to run the tests. The majority of them will be exploratory testing guided by the test scenarios. Using this method allows the largest number of defects to be found in the shortest possible time.

When the tests are performed by the operators and users, they use the test cases. These testers execute each test and compare the results to the expected results, assigning a pass, fail, or skip rating. The fail rating requires the tester to document what really happened during the test. In this phase, bugs are also logged into the bug tracking system identified in the test plan. Once the development team fixes the defect, the test case can be rerun in accordance with your test plan.

6. Test cycle closure

This is the final phase of the STLC where upon execution of all test cases, the QA lead confirms that all required testing has been completed. Here, an analysis is conducted of the test cases - how many passed, failed or were skipped, defects found, and other metrics. This creates an opportunity for the team to improve future testing projects.

Which type of software testing is best for you?

Many testing teams are turning towards automation to increase test coverage and efficiency, and some still wonder whether they should make the switch. However, before this decision can be made, you first need to determine what you need to achieve. Ultimately, automation should save you money, time, and effort. Automating some tests could prove to be a more costly solution than an economic one, so to help you decide, we have provided a few criteria to be taken into consideration.

1. Functional test cases

Functional testing is the process through which quality assurance is achieved by validating that the software system meets its functional specifications. Automated testing makes this process quick and seamless.

2. Repetitive test cases

It would be senseless to use automation for one-off tests, however, test automation is a must if there are repetitive tests that need to be executed on demand. It will reduce the cost per test run, as well as the completion time of a development cycle.

3. Critical test cases

Some test cases pose a high risk to the business in terms of cost, user experience, and customer satisfaction. In such cases, it would be preferable to use automated testing, because, with manual tests, even the most experienced testers can produce error-prone codes.

Conclusion

To achieve superior quality software, a robust testing system is imperative. Software testing allows you to verify each and every aspect of the software application, giving ample opportunity to fix any errors that may present. Bear in mind, however, that to be truly effective, best practices must also be incorporated into the mix.