What Are the 7 Principles of Test Automation?

Software testing has assumed significance to make the software glitch-free and more amenable to users. Software testing has become an integral part of the Agile-DevOps model of software development. Given the inadequacies of manual testing in validating repetitive and voluminous codes, test automation has assumed salience. It helps reduce testing efforts and time, especially when the testing involves multiple devices, platforms, and environments. Any enterprise process automation is guided by a few principles to make the activity worth the effort. The 7 principles guiding any test automation activity are discussed below.

The Top 7 Principles of Test Automation

The objective of test automation services is to ensure the build to be tested is stable, risk-free, and clear of any glitches. Using a minimal number of resources, test automation can achieve results that are not possible to conceive let alone achieve. The 7 principles to guide test automation are:

1.Improve quality: As a concept, quality cannot be defined but experienced. This is especially true when users use a software product, wherein the presence of glitches can mar that experience. There are plenty of metrics to measure quality, such as code coverage, number of defects, test failure rate, CI error rate, and others. Each of these reflects the quality of a

software application in some aspect. Test automation can improve these metrics by running tests iteratively and finding defects in the process. This way, stakeholders can understand whether the build is ready to be released into the market or whether it meets all expectations. Moreover, development driven by tests in the form of Test-Driven Development (TDD) and Behavior-Driven Development (BDD) can lead to success.

2.Minimize risks: It is a fact that peer code review with more eyeballs does not find all the bugs present in a piece of code. The best way to achieve that is by building and running an automated test suite. The suite can run the application across devices, platforms, and environments to identify errors or bugs. It can do so at a scale that is not possible manually. Since software applications have third-party dependencies, errors can creep in with associated risks. Thus, software test automation services can check every line of code repeatedly for errors and reduce risks for any software application.

3.Context-based testing: Test automation is not a “one size fits all” thing. It needs to be customized based on the features, elements, and requirements of a product or project. For instance, banking software would need more rigorous and comprehensive testing than, say, software for entertainment. So, no two tests can have the same approach.

4.Know about the system: Developers can sometimes found themselves to be at sea dealing with codes written by someone else. They can even forget how the code works if they happen to take a hiatus. So, the best way to acquaint oneself with the code or application in the making is by referring to the automated test suite. The suite can shed light on various aspects of the application and its functionality, such as how the API will respond, among others.

5.Easy to write test cases: Best test outcomes can only come about when there is a simpler and granular code. Writing complicated codes for test suites can create the risk of failure, leading to false positives. Writing simple test cases can follow the below-mentioned principles:

- Test and build should be separate
- Tests should be independent of each other because otherwise dependencies could get complicated
- Do not use similar code twice and avoid test overlap
- Test external interfaces rather than break test code encapsulation

6.Easy test run: Tests should run without any human intervention for every code change. This means that tests should be granular, lightweight, independent, deterministic, and display

idempotency. Further, the CI/CD pipeline should act as a quality gate wherein instant feedback can be obtained for every test run.

7.Low maintenance: An important part of writing a test code is to maintain it for further use. The ways to ensure low maintenance include writing adaptive test scripts, changing tests for apps, taking a clear data management approach, automating at the API level, and conducting regular health checks, among others.

Conclusion

The costs of avoiding test automation are many, including a negative impact on brand reputation. Even if the impact of glitches is not apparent on the value of the product, it can spiral out of control if not nipped in the bud. By implementing software QA automation, businesses can lay the foundation to develop and deliver quality software. It is better to hire an experienced automation testing company to manage the entire test automation cycle.