Software Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Some prefer saying Software testing definition as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test (AUT). This Software Testing course introduces testing software to the audience and justifies the importance of software testing. **Software Testing is Important** because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction. As per ANSI/IEEE 1059, **Testing in Software Engineering** is a process of evaluating a software product to find whether the current software product meets the required conditions or not. The testing process involves evaluating the features of the software product for requirements in terms of any missing requirements, bugs or errors, security, reliability and performance. This is not the complete list as there are more than 150 types of testing types and still adding. Also, note that not all testing types are applicable to all projects but depend on the nature & scope of the project. **Unit Testing:** This software testing basic approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not. **System testing:** In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others. **Program Testing** in software testing is a method of executing an actual software program with the aim of testing program behavior and finding errors. The software program is executed with test case data to analyse the program behavior or response to the test data. A good program testing is one which has high chances of finding bugs. There are numerous types of software testing techniques that you can use to ensure changes to your code work as expected. Not all testing is equal though, and we explore how some testing practices differ. It's important to make the distinction between manual and automated tests. Manual testing is done in person, by clicking through the application or interacting with the software and APIs with the appropriate tooling. This is very expensive since it requires someone to setup an environment and execute the tests themselves, and it can be prone to human error as the tester might make typos or omit steps in the test script. Automated testing is a key component of continuous integration and continuous delivery and it's a great way to scale your QA process as you add new features to your application. But there's still value in doing some manual testing with what is called exploratory testing as we will see in this guide. Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software. Unit tests are generally quite cheap to automate and can run very quickly by a continuous integration server. Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running. End-to-end testing replicates a user behavior with the software in a complete application environment. It verifies that various user flows work as expected and can be as simple as loading a web page or logging in or much more complex scenarios verifying email notifications, online payments, etc... End-to-end tests are very useful, but they're expensive to perform and can be hard to maintain when they're automated. It is recommended to have a few key end-to-end tests and rely more on lower level types of testing (unit and integration tests) to be able to quickly identify breaking changes. Acceptance tests are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors. But they can also go further and measure the performance of the system and reject changes if certain goals are not met. To automate your tests, you will first need to write them programmatically using a testing framework that suits your application. PHPUnit, Mocha, RSpec are examples of testing frameworks that you can use for PHP, Javascript, and Ruby respectively. There are many options out there for each language so you might have to do some research and ask developer communities to find out what would be the best framework for you. An exploratory testing session should not exceed two hours and should have a clear scope to help testers focus on a specific area of the software. Once all testers have been briefed, various actions should be used to check how the system behaves. To finish this guide, it's important to talk about the goal of testing. While it's important to test that users can actually use an application (they can log in and save an object), it is equally important to test that an application doesn't break when bad data or unexpected actions are performed. You need to anticipate what would happen when a user makes a typo, tries to save an incomplete form, or uses the wrong API. You need to check if someone can easily compromise data or gain access to a resource they're not supposed to. A good testing suite should try to break your app and help understand its limit. I've been in the software business for 10 years now in various roles from development to product management. After spending the last 5 years in Atlassian working on Developer Tools I now write about building software. Outside of work I'm sharpening my fathering skills with a wonderful toddler. Software testing is the process of assessing the functionality of a software program. The process checks for errors and gaps and whether the outcome of the application matches desired expectations before the software is installed and goes live. Software testing is the culmination of application development through which software testers evaluate code by questioning it. This evaluation can be brief or proceed until all stakeholders are satisfied. Software testing identifies bugs and issues in the development process so they're fixed prior to product launch. This approach ensures that only quality products are distributed to consumers, which in turn elevates customer satisfaction and trust. To understand the importance of software testing, consider the example of Starbucks. In 2015, the company lost millions of dollars in sales when its point-of-sale (POS) platform shut down due to a faulty system refresh caused by a software glitch. This could have been avoided if the POS software had been tested thoroughly. Nissan also suffered a similar fate in 2016 when it recalled more than 3 million cars due to a software issue in airbag sensor detectors. There are many types of software testing, but the two main categories are dynamic testing and static testing. Dynamic testing is an assessment that's conducted while the program is executed; static testing examines the program's code and associated documentation. Dynamic and static methods are often used together. Over the years, software testing has evolved considerably as companies have adopted Agile testing and DevOps work environments. This has introduced faster and more collaborative testing strategies to the sphere of software testing. Automated testing can be used to test larger volumes of software when manual testing becomes tedious and time-consuming. Test scripts can be run automatically on software applications, which frees up time and resources and enables companies to test efficiently at lower costs. Many QA teams build in-house automated testing tools so they can reuse the same tests repeatedly and deploy them around the clock without time constraints. Most vendors also offer features for streamlining and automating tasks. For automated testing of web application frameworks, tools such as Java for Selenium are often used. There's more to software testing than running multiple tests. It also entails using a specific strategy and a streamlined process that helps to carry out these tests methodically. To improve the performance and functionality of any application or product, software best practices should always be followed. Computer scientist Tom Kilburn wrote the first piece of software code in 1948 at the University of Manchester in England. Software testing started during the same timeframe but was restricted to debugging only. By the 1980s, development teams started to incorporate a more comprehensive process for isolating and fixing bugs and doing load testing in real-world settings. This brought software testing to the forefront. In the 1990s, the QA process was born and testing became an integral part of the software development lifecycle. Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, QA Analyst, etc. **Cost Effective Development** - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.

Download

Software Testing

21f597057a