I'm not a bot

Software testing is the process of assessing a software application to evaluate its functionality and verify that it meets user expectations. Well review the most common types, compare different methodologies, and learn the importance of testing in software development.Common Types of Software TestingIn software engineering, testing occurs in regimented steps meant to check different aspects of the application, including functional, nonfunctional, and maintenance capacities.Functional TestingFunctional testing is a quality assurance (QA) process evaluating whether the basic capabilities of the software meet specified requirements.Unit Testing: A developer tests individual functions through tools like Jest, Vitest, and Playright. After youve established workflows, a continuous integration service like GitHub can automatically conduct unit and integration tests.Integration Testing: The developer assesses how software components operate together. The objective is to see how an individual code module integrates with the entire system. For reference, one software application is generally composed of several modules built by different programmers.System Testing: Professional QA testers evaluate the whole software system for quality assurance before introducing it to the market.User Acceptance Testing: The final testing phase involves end users beta testing the software product. Assessments target specific expectations and requirements established by consumers. Nonfunctional TestingNonfunctional testing assesses software properties that are non-critical to functionality but imperative for a positive user experience.Smoke Testing: Also known as confidence or sanity testing, this preliminary test evaluates a systems most critical functionalities to reveal whether they are stable enough to proceed to quality assurance.Performance Testing: Engineers or performance testers evaluate software responsiveness, speed, and stability under different workloads.Usability Testing: Beta testers help assess software accessibility by measuring how successfully they complete typical tasks.Stress Testing: Developers test how much strain the software can handle under abnormal conditions and analyze recoverability and stability after a system fails.Load Testing: Developers test a systems capacity by placing a simulated load or demand on the web application.Maintenance TestingAlso known as post-release testing, maintenance testing involves fixing bugs and releasing new features to a live application.Confirmation Testing: Developers evaluate whether previously identified bugs are fixed or not.Regression Testing: Developers assess whether an application behaves as expected after code updates or changes.Testing Techniques and MethodologiesThere are multiple approaches for software testers, depending on the objectives they wish to achieve.Static vs. Dynamic TestingStatic testing evaluates software without executing the code and checks for flaws and errors. Dynamic testing evaluates functional and nonfunctional aspects of the software while executing code in a run-time environment.Static TestingDynamic TestingAnalyzes software code and documentationAnalyzes software functions, bugs, bottlenecksPrevents defectsIdentifies defectsPerformed before code compilationPerformed after code compilation In automated testing methods, developers write test scripts created within software development tools to assess the application. Manual test strategies are different and require a testing team of QA analysts to identify bugs before applications go live.Automation TestingManual TestingUtilizes test scriptUtilizes human resourcesCost-effectiveMore expensiveFaster processSlower processIncludes regression, stress, performance testingIncludes ad-hoc, usability, exploratory testingGray, White, and Black Box TestingGray, black, and white box testing refer to three methods conducted from different perspectives, including those of QA testers, developers, and end users. These testing processes must operate in conjunction with one another during software development.Black Box TestingWhite Box TestingGray Box TestingNo internal knowledge of softwareInternal knowledge of softwarePartial knowledge of softwareEvaluates functionalityEvaluates structureEvaluates bugs and vulnerabilitiesPerformed by testers and end usersPerformed by developersPerformed by developers and testersRequires no programming knowledgeRequires extensive programming knowledgeRequires basic programming knowledgeImportance of Software TestingSoftware quality assurance can identify costly mistakes before the application deploys, pinpointing scalability issues, security vulnerabilities, or broken functionalities.For example, in 2015, Starbucks suffered a massive shutdown of its point-of-sale platform. Failure during a daily system refresh disabled cash registers in stores across the United States and Canada, more than 60% of 13,500 locations.National Grid, an East Coast power distribution company, only tested its new ERP system in best-case conditions. After Hurricane Sandy struck in 2012, the software revealed its inability to coordinate with hundreds of external utility vendors needed for power restoration. As a result, National Grid sued system integrator Wipro for $75 million but ultimately lost $585 million due to failed implementation.By addressing poor design choices, performance issues, and architectural flaws from the beginning, software development companies can save millions of dollars annually and increase their market share.BenefitsThe primary goal of software testing is to diagnose an applications errors and flaws. There are four main benefits to testing during and after development:Cost-effectivenessA development team can utilize software testing to reduce costs in the long run, minimizing the need for frequent software updates.Customer satisfactionFrom the jump, you can give your clients a better, bug-free experience and a more responsive user interface. Building customer loyalty through a high-performing product will only grow your client base.SecurityReduce potential exploits through frequent security testing. This ensures that vulnerabilities and weaknesses in your application do not comprise company and/or user data.Product qualityEnsure your end product meets requirements through a rigorous quality assurance process. By focusing your testing efforts on the system, user interface, and API fronts, youre putting your software in the best position for success come launch day.FAQsWhat is a test case?A test case is a set of execution conditions and expected results developed around a test scenario. Analysts follow these test plan instructions to evaluate if the software performs a function correctly.How does verification testing differ from validation testing?Verification testing evaluates whether your software meets established requirements. Validation testing assesses whether your software meets customers expectations and needs.What are the different types of software testing?Software testing covers three broad categories:Functional testing: Evaluates whether the basic capabilities of the software meet specified requirements.Non-functional testing: Assesses software properties that are non-critical to functionality but imperative for a positive user experience.Maintenance testing: Involves fixing bugs and releasing new features to a live application. Search Networking Search Security Search CIO Search HRSoftware Search Customer Experience Software Testing is a process used to identify bugs or errors present in software applications during the software development lifecycle. Many software organizations follow theAgiledevelopment methodology, which includes software testing in each iteration. Software Testing is done in two different ways as Manual and Automation. This categorization happens based on the objective of the testing, tools, and methods used by testers. The process of automation testing is performed with automation tools. This article discusses the concept of software testing, types of testing, principles, best practices, and, benefits of software testing.What is Software TestingSoftware testing is the process of evaluating the functionality of a software system or application. The main objective of software testing is to compare the actual function results with the expected functionality to ensure that the software application is free of defects and ready for use. It involves identifying product bugs and requirements that have not been met, and this is the core concept discussed in this document.History of Software TestingThe computer scientist, Tom Kilburn was the first person to write a piece of software code to perform mathematical calculations and launched it on June 21, 1948. At that time only Debugging was considered the main testing method and it continued to remain the same for the next two decades.In the 1980s, the first version of the IEEE 829 Standard for Software Test Documentation was published, which motivated software testers to pay attention to the quality assurance of the software product. From the 1990s, it started to improve more on methodologies and Software testing tools that were going to be used in the software testing process. In recent years, software testing has seen enormous progress with the introduction of powerful automation tools like Selenium, Robotium, Testsigma, etc.Read here Software Testing MethodologiesImportance of Software Testing: Why is it Needed?Every software development process aims to deliver a quality product to its users, by satisfying all the business requirements. Software testing can be identified as the peek point of such a development process, as it continues until the majority of the bugs and issues are found. Following are some of the reasons to showcase the importance of software testing.Identify the poor design and architectural level decisions at early stages.Software testers who participate in every testing phase gain a deeper understanding of the project, identify issues earlier, collaborate with developers, and produce a higher-quality final product. This improves the efficiency of the development process, reduces the risk of negative feedback, and ultimately leads to greater user satisfaction.Improve the quality of the final product.Principles of Software TestingTo get the best test results, in the end, this testing process should be conducted properly. For that reason, a set of principles are being followed in the software industry.Exhaustive testing is not possibleIn Exhaustive testing, specific functionality is tested using all possible combinations of inputs (valid and invalid). If the software is tested at every test case, then it may cost more money and hard work and, it becomes time-consuming as well. As it seems impractical to follow, only an optimal number of test cases are executed, and based on their result the software product is considered correct.It has been identified that, in a project, a small number of modules can contain most of the defects. This is based on the Pareto Principle in software testing: 80% of software defect comes from 20% of modules.Executing the same test cases many times will make it lose the ability to trace new bugs or defects in the software. Therefore, a time-to-time review of the test cases should be done to find more new bugs.Testing shows the presence of defectsSoftware testing always points out the defects in the application. It never discusses or claims zero defects. Even after running multiple test cases, the application cannot be considered bug-free. Therefore, testing can only reduce the number of bugs but, cannot eliminate all of them.Absence of error the fallacyIf the software is tested as 99% free but, still does not meet the customers requirement, then the product is considered unusable. Therefore it cannot claim that a 99% bug-free application must also meet the customer requirements.To detect defects at the earliest, it is important to do testing from the first phase of SDLC. Testing that is performed during the requirement analysis phase (the first phase of the SDLC) may cause to increase its performance.Testing is context-dependentBased on the context of the developed software, the testing approach is chosen for testing. The same testing approach may not be suitable for all types of software.What Are the Benefits of Software Testing?Following are some of the core benefits of software testing,Ensures to deliver a quality product the product is thoroughly tested before handing it over to the customer. Software testing ensures that the product meets all the requirements set by the customer.Identifies bugs and defects early software bugs can be expensive or even dangerous. In the development cycle, it identifies bugs before launching the application to use.Long-term cost saving. Software testing saves money by catching issues early, preventing costly recalls, and dissatisfied customers. It also reduces the cost of fixing defects later in the development process or after release, resulting in long-term cost savings. Read more- Cost of Defects.Customer satisfaction and trust. Software testing enhances customer satisfaction and trust by ensuring that the final product is of high quality, meets user requirements, and protects user data. This results in positive user experiences, feedback, and stronger relationships between users and the software company.Security A well-tested software application prevents cyber-attacks. For example, insecure application code can give access to the customers browser by allowing the injection of malicious code.Different Types of Software Testing TechniquesThe purpose of doing software testing is to identify and eliminate the majority of the bugs in a Software application. To handle the testing process efficiently and to cover all the possible test cases at a specific development level, a set of software testing types are followed. Black Box testing and White Box testing are two major categories in software testing.White-box testing is a software testing technique that examines the internal structure, design, and code of an application to ensure its functionality, security, and adherence to specifications. Thorough testing and execution of each line of code are essential in the software development process. Before the application is handed over to software testers, white-box testing is typically performed. This type of testing can be automated. Usually, this testing is conducted as part of the initial unit testing process. Ultimately, who performs the testing depends on the preferences of the team or the specific needs of the project.. Black-box testing is a software testing technique conducted without prior knowledge of the internal workings of the application, focusing on validating its functionality and behavior based on the expected input-output scenarios. If there are defects found in this testing, the application will be sent back to developers for fixing. At the end of each black box testing, it is ensured that the specified functionality meets the business requirement. This is normally performed by testers who are not well-versed in coding. Therefore, this requires automation.This testing process is a collaboration of both black-box and white-box testing. Its purpose is to find defects in the code structure and unsatisfied requirements in the functionality. The individual performing the grey box testing possesses knowledge of coding and testing, which allows them to effectively conduct the testing process, potentially combining manual and automated techniques as needed. Therefore even though test automation is possible for Grey box testing its not mandatory.Black box testing is further divided into functional and non-functional testing.The applications functionality is checked to verify whether they are working as specified in the requirement. Each functionality is checked by providing appropriate inputs, verifying the output, and comparing the actual results with the expected results. This type of testing can either be automated or done manually. It is better if it is automated because it makes the testing process quicker.Individual units/components of a software application are tested. The purpose of this testing method is to validate that each unit of the software performs according to the design.Can they be automated? Yes, unit testing can be automated.Should they be automated? Yes, automating unit tests can save time and increase efficiency.This type of testing validates the fully-integrated software application.Can they be automated? Yes, some aspects of system testing can be automated.Should they be automated? It depends on the specific system being tested and the goals of the testing process.Individual modules of an application are combined and tested to verify if they are working as they intend to when integrated. This type of testing assumes that units/modules have been tested individually. Integration testing can be done by using two approaches.Can they be automated? Yes, integration testing can be automated. Should they be automated? Yes, automating integration testing can save time and improve accuracy.I) Big-bang ApproachAll the modules are tested together at once and then tested as a unit. As all the components are merged at once, it is difficult to identify the cause of a failure. This approach is more suitable for smaller systems.Ii) Incremental ApproachUnder this approach, top-down, bottom-up, and sandwich methods are applied.Starts by testing the top-most modules and gradually moving down to the lowest set of modules. It identifies major problems early. Also, the test cases can be defined in terms of functional requirements.Starts testing the lower level units and gradually moving up one by one. Higher-level modules might not have been developed by the time the lower modules are tested.This testing approach is a combination of both top-down and bottom-up integration. The application is viewed as a layer.This type of testing focuses on the non-functional requirements of an application. This type of testing always remains automated since this checks non-human interactions.Usability testing measures, how easily the system/application can be used.Can they be automated? No, usability testing is typically conducted manually.Should they be automated? No, since usability testing involves subjective evaluation, it is best conducted manually.This ensures the application is capable of running on different hardware, operating systems, network environments, etc.Can they be automated? Yes, compatibility testing can be automated. Should they be automated? Yes, automating compatibility testing can save time and increase efficiency.There are two ways of doing compatibility testing,Forward compatibility testing not like backward compatibility testing, this method verifies the behavior of the developed application with the newer versions of the application.This type of testing checks the performance objectives of the application (such as speed, response time, reliability, etc.) under their expected workload. The focus is to eliminate performance bottlenecks in the software while measuring the performance of each component.Verifies if the application is able to perform under expected user workloads.Identifies the performance bottlenecks.Ex: Testing a printer by sending a large job continuouslyReading and writing data into the hard disk.The application is tested under extreme workloads to see how high traffic is handled by the applicationIdentifies the applications breaking pointDetermines failures of a software system and Identifies how thesystem recovers from its failures. (Recoverability)Can they be automated? Yes, stress testing can be automated.Should they be automated? Yes, automating stress testing can save time and improve accuracy.Tests the softwares reaction to sudden large spikes in the load generated by users.Can they be automated? Yes, spike testing can be automated.Should they be automated? Yes, automating spike testing can save time and increase efficiency.To ensure that the expected load can be handled by the application for a long period of timeCan they be automated? Yes, endurance testing can be automated.Should they be automated? Yes, automating endurance testing can save time and improve accuracy.Determines the applications effectiveness in scaling up to support an increase in user load.Can they be automated? Yes, scalability testing can be automated.Should they be automated? Yes, automating scalability testing can save time and increase efficiency.Monitors system behavior by inserting a large number of data into theDatabase.Can they be automated? Yes, volume testing can be automated.Should they be automated? Yes, automating volume testing can provide more accurate and consistent results.Read all about Levels of Testing.Software Testing ModelsVarious software testing models guide the testing process and define its phases. Some common models include:V-model: This model follows a V-shaped process, with each testing phase linked to its corresponding development phase.Waterfall model: Similar to the V-model, testing occurs sequentially after development phases are completed.Agile model: Testing is integrated throughout the development process in short iterations.Spiral model: Combines elements of iterative and sequential models.Test Pyramid model: Emphasizes a balanced testing approach with more unit and integration tests than UI tests.The choice of model depends on the specific project requirements and team preferences.When to Go for Automation in Software Testing?A test automation tool (Testsigma, JUnit, Jmeter, Selenium, postman, etc.) is used to execute test cases in automation testing. It is less error-prone compared to the manual testing process. Automation testing may be suitable in the following situations,When tests need to be executed frequently.When tests are too complex to be executed manually.When test execution time needs to be reduced.When the testing team is small and test automation needs to be put in to save money.When test coverage needs to be improved.What Can Be Automated within Software Testing? Many aspects of software testing can be automated, including:Repetitive tasks: Automating repetitive tasks like regression testing can free up tester time for more complex tasks.Data-driven testing: Automated testing can be driven by data to test different scenarios and configurations.UI testing: Tools can be used to automate interactions with the user interface.API testing: Tools can be used to automate tests for application programming interfaces (APIs).Performance testing: Tools can be used to automate performance testing tasks such as load testing and stress testing.While automation plays a vital role in modern software testing, its essential to remember that it cannot replace manual testing entirely. .How to Automate Your Software Testing with Testsigma?Testsigma is a cloud-based automation tool that enables creating tests for web apps, mobile apps, and APIs within a short span of time. Testsigma offers a free trial and two paid plans pro and enterprise.The following steps show how a web application test can be automated with Testsigma.Create a Test case.Add Test steps in the test case that was created in Step 2. There are two options to add test steps.This example shows the steps to follow the first option.Navigate to the web application that needs to be tested. Paste the URL of the web application in the Test Steps and, click Create.Click the Record button to start the recording. This will navigate to the web page you have given and you can perform any action, and it all will be recorded.Click the Finish button and, it will navigate to the Test case detail page that has been created using the recorded actions.The following screenshot shows the test case taken from the recording and, once the Run button is clicked, the result sheet will be displayed.Automate your tests for web, mobile, APIs, and desktop applications, 10x faster, with TestsigmaWhat If There is No Software Testing in the Software Development ProcessThe objective of software testing is to find if there are any defects in the software, so that they can be removed or reduced, and also, to check whether the software functionality meets the customers requirements. Only a product that is with fewer errors and that satisfies all the customer requirements can be accepted as a quality product. To obtain a quality product, it should be tested thoroughly before handing it over to the customer. Following are some of the possible risks when software testing is not present.In the Agile environment, it is possible to detect bugs at early stages, which requires a lesser amount of money to fix. If software testing is not done, defects would be reported by customers after the software is already released. This situation can cause you to pay a large amount of money, including long working hours, increased downtime, and loss of productivity.Lose organization reputationA user with a bad experience may tend to pass negative feedback to others. This also leads a customer to have a negative view of the product as a whole, which in turn may break the trust of the organization.Lose customer satisfaction.The final product should fulfill all customer expectations. A user with a bad user experience may lose interest in the product and the organization. Therefore it is important to test a software application in a proper agile environment.Software Testing Fundamentals: Understand Basic Concepts and ProcessesSoftware testing fundamentals involve identifying and resolving issues in the software. It ensures that the software is reliable, efficient, and usable.But how do we do that? By breaking the software testing process into basic components, including the importance of software testing, its levels, the different types, testing techniques, and test methods.At a deeper level, software testing basics also include the impact of bugs or errors in the code, how to overcome them, the causes of software defects, and their consequences.The guide covers all the software testing fundamentals, such as:Types of software testingPrinciples of software testingIts benefitsTesting modelsAutomating all these processes using TestsigmaBest practicesYou can explore this software testing strategy guide for a deeper dive into the strategies behind these fundamentals.Objectives of Software TestingIf software testing is like preparing a meal for a special occasion, what are our primary goals or objectives for this preparation? Why do we put in all this effort? Here are the direct objectives of software testing:Ensure Reliability: Just as youd want your dish to be consistently delicious every time you prepare it, we test to ensure the software is dependable and wont disappoint you when needed.Verify Functionality: Its like ensuring all the ingredients in a recipe come together ideally. In software, we verify that every feature functions as intended.Identify Defects: Before serving a dish, youd want to ensure it tastes right and has no unwanted elements. Similarly, testing helps spot and fix any issues or bugs in the software. Its essential to be aware of common pitfalls, as highlighted in our article on software testing errors to look out for.Assess User Experience: Just as the presentation and taste of a dish enhances your dining experience, a user-friendly software interface ensures a delightful digital journey.Maintain Quality Standards: Just as dishes must meet specific culinary standards, the software has quality benchmarks it needs to meet.Consequences of Not Performing Software TestingImagine preparing a meal for guests without tasting it first. Sounds like a recipe for disaster, right? Just as that meal might not turn out as expected, skipping software testing can lead to some significant issues. If we dont test the software, heres what could happen:Frequent Breakdowns: Untested software can crash or freeze, disrupting your tasks and causing frustration, much like serving a dish thats too salty or undercooked.Security Risks: Without testing, the software might have vulnerabilities that hackers can exploit, putting your data at risk, akin to leaving food out in the open and attracting pests.Increased Costs: Addressing software problems after release can be costlier than catching them during testing before release, just as fixing a ruined dish at the last minute can be time-consuming and wasteful, leading to additional ingredients or even an entirely new preparation.Poor User Experience: Untested software might have glitches or design flaws that make it hard to use, similar to trying to enjoy a meal that is missing key ingredients.Damaged Reputation: Buggy software can harm a companys reputation and user trust, much like a chef known for inconsistent dishes might lose customers.Check here Canary TestingSoftware Testing Best PracticesRunning test cases several times wont be enough to perform software testing successfully. Following are some of the practices that can improve the software testing process.A definition of the expected output or result is a vital aspect of a test case. If the expected result of a test case has not been predefined, an error-prone but plausible result can be labeled as the correct one.A developer should never test his or her program. This is to test the program with different perspectives. If the developer has misunderstood the problem specification, then the same misunderstanding will carry into tests of their program.An organization that is into programming should not test programs that belong to them.A thorough analysis of each test result is a best practice in software testing. This involves carefully reviewing and interpreting the results of each test to identify any issues or potential areas for improvement. This helps ensure that any defects or problems are caught early and addressed appropriately, improving the overall quality of the software being developed.Test cases written should consist of both unexpected and invalid. At the same time, scenarios that are expected and valid should also be included.Negative testing is a best practice that involves intentionally testing the software with invalid or unexpected inputs to identify potential vulnerabilities or weaknesses. This helps ensure that the software is robust and can withstand a variety of scenarios.Test cases that have a low value and consume a high cost to execute should be avoided.Read here about QA Process.Applications of Software TestingSoftware testing is used in various applications, including:Web applications: Testing ensures web applications function correctly across browsers and devices.Mobile applications: Testing ensures mobile apps function smoothly on different devices and operating systems.Desktop applications: Testing ensures desktop applications function properly and meet user requirements.Embedded systems: Testing ensures embedded systems function reliably within their intended environments.Gaming software: Testing ensures games are enjoyable, bug-free, and provide a smooth gaming experience.ConclusionIt has been proven that software testing is vital in the process of software development. A software product that meets all the customer requirements and is thoroughly tested will have better quality. Selecting the correct testing methods and tools while following testing principles can ease the process of software testing. Results of tests you had at your GP surgery, and results that have been shared with your GP surgery, may be available in your online GP health record.View test results in your GP health record Testing plays a crucial role in the software development life cycle, ensuring that applications function properly and meet stakeholders expectations. Software testing is the process of evaluating and verifying that a software application or system meets its requirements and functions as expected. It involves testing to identify bugs or defects and ensure quality.Types of Software Testing:Functional Testing verifies that the software functions as per the requirements. Examples include Unit Testing, Integration Testing, System Testing, and Acceptance Testing.Non-Functional Testing evaluates the softwares performance, usability, and other non-functional aspects. Examples include Performance Testing, Security Testing, Usability Testing, and Compatibility testing.Software Testing Life Cycle:Requirement Analysis: Understand and analyze test requirements.Test Planning: Define the scope, strategy, and resources for testing.Test Design: Create test cases based on requirements with expected outcomes.Test Environment Setup: Set up the necessary hardware, software, and network configurations.Test Execution: Run test cases, record results, and identify defects.Test Closure: Evaluate results, prepare reports, and finalize the testing phase.Benefits of Software Testing:Ensures product quality and meets user expectations.Reduces maintenance costs by identifying defects early.Improves user satisfaction by providing reliable and secure software.Enhances software performance and usability. What is Software TestingSoftware Testing is a process of evaluating the functionality of a software under test. This process helps in verifying and validating the actual software with the expected requirements to ensure that the product is built in the right way and is defect free. This process is not just restricted to finding faults and bugs in the software but also to improve software efficiency, accuracy and usability.As defined by the ANSI/IEEE 1059 standard,Testing is the process of analysing a software item to detect the differences between existing and required conditions (that is, defects/errors/bugs) and to evaluate the features of the software item.The prime objective of Software Testing is to find issues early in the development phase of SDLC (Software Development Life cycle) to minimise the risks at the later stage. Software testing can be achieved manually or through automation. While manual testing involves a tester to examine software manually, automation testing is performed by writing test scripts using any preferred programming language and executing them in order to evaluate the software against the requirements.Read More: How to learn Software Application TestingSoftware Testing Life Cycle (STLC)Software Testing Life Cycle (STLC) is a structured approach to assess the quality and functionality of the software application to ensure that it meets the requirements and is bug free. Each phase in STLC has specific objectives and deliverables. The prime objective of STLC is to find and document any defect in the software application as early as possible at the initial stage to minimise the risk of fixing at the later stage. The stages of STLC are as follows:1. Requirement Analysis: At this first stage, testers review the requirements and understand what needs to be tested. They identify any missing requirement and report it to the stakeholder. If there is any ambiguity or difficulty in understanding the requirements, they meet the stakeholder to clear the confusion. These activities help testers create better test plans.2. Test Planning: This is the most crucial phase of STLC as all the testing plans are defined at this stage. Once the requirement analysis is completed; the test manager carries out the following tasks:Identify the objective and scope of testing.Develop a test strategy that includes testing methods and techniques to be used.Identify and analyse the testing environment and resources required.Identify test cases and test data to be used.Calculates the estimated time, effort and cost of the testing.Allocate roles and responsibilities to the test team.Review and approve the test plan.3. Test Case Design: This phase involves creating detailed test cases that specify the steps to be executed, pre-conditions (if any), test data, and expected result for each scenario. Following are the activities performed while designing test case:Identifying the test cases based on the test scenarios.Developing test cases which are simple, easy to understand and detailed. Details should include preconditions, steps to perform, test data and expected outcome. Test cases should be extensive and should cover all the possible cases.Reviewing, updating and approving the test cases.Finally, create a Requirement Traceability Matrix (RTM) to map requirements to the test cases.4. Test Environment Setup: To execute the test cases testing team needs an environment (installing servers, hardware/software) which should be more or less similar to the environment used by the end user/customer. Mostly the environment is set up by the developer with supervision of the Test Manager. This is an indispensable phase as it decides the conditions on which software needs to be tested. A quick smoke testing should be performed as soon as the environment is developed to check the readiness.5. Test Execution: Post test cases development and environment setup, now it is time to do the actual business that is executing the test cases and publishing a report to showcase the health of the Software application.Following are the activities carried out by the testing team:Test cases created during the previous stage are executed to surface any failures.Actual Result is compared with the expected outcome and any defect found during the execution is logged in the defect tracking management with necessary details such as steps to reproduce, severity, priority, logs, etc.Developers review the defects reported, fix it and then assign it to the testing team to retest the same.Testing team retests the defect and closes it in case it is not reproducible and reopens it, if it is still resurfacing. Regressing testing is performed to ensure that the changes made to fix defects do not introduce new issues.Test reports are created and shared to the relevant audience.6. Test Closure: This is the final stage of STLC which involves formal conclusion of testing activities after all the test cases have been executed, bugs reported and resolved, and test exit criteria met. The prime objective of test closure is to verify that all the activities related to testing have been completed and the software is in the best form to be released. Importantly, test closure should also include test process documentation.Following activities are included in test closure:Test Summary Report: A test report is created which include all the vital information pertaining to the test execution such as number of test cases executed, pass/ fail/ skip statistics, testing effort and time metrics, number of defects found, resolved and closed, any open issues which goes as known defect in production, etc. This serves as a record to all the stakeholders which can be referred by anyone in future.Defect management: All the issues identified during execution are tracked and maintained in the defect management tool and accessible to all the stakeholders.Evaluating exit criteria: Exit criteria defined in the test plan is revisited and reviewed by the testing team to ensure that all the criteria have been fulfilled before releasing the software in production.Feedback/ Improvements: Any challenges or bad experience encountered during the course of testing is analysed and resolved. The feedback gathered by the testing team is worked upon and improved to smoothen the future testing process.Sign-off: Lastly, the test closure is signed-off formally by the project managers and other key stakeholders to signify that the testing phase has been completed satisfactorily and is meeting the defined criteria. It addresses that the testing activities have been reviewed and approved by them and the software is ready to be launched on production.Read More: Software Testing Strategies and ApproachesSoftware Testing ToolsThere are several Software Testing tools available in the market to assist the testers in all the types of testing such as Functional, Performance, Test Management, Compatibility tools, etc. Below are some of the free tools available in the market which are very popular and loved by most testers.BrowserStack: A popular cloud-based testing platform that simplifies testing by providing access to 3500+ real devices and browsers. BrowserStack offers Live for manual testing, Automate for automated testing, and Percy for visual testing. With seamless integration and scalability, it empowers teams to accelerate development and deliver high-quality software.Selenium: An open-source automation framework used to automate web applications and supports various programming languages such as Java, Python, C#, JavaScript, etc.Appium: An open-source automation framework for testing mobile native, hybrid and web applications. Like Selenium it also supports various programming languages and can be used for Android and iOS both.Cucumber: A behaviour-driven development (BDD) technique which helps to collaborate and create executable specifications written in a human readable format. Cucumber uses the Gherkin language which is a plain-text and easy to read language and uses keywords like Given, When, Then, to define test scenarios and steps.Katalon Studio: An automation testing tool that supports web, mobile, desktop applications and API testing. It can be easily run on Windows, macOS and Linux operating systems. Unlike other automated testing tools, testers with very less programming skills can also use it.Test Complete: An automation testing tool used to test desktop, mobile and web applications. It supports programming languages such as JavaScript, VBScript and Python and testing techniques such as data-driven, keyword-driver and regression.Jira: Most popular bug tracking tool which is mostly used for issue tracking, test planning and test management.JMeter: A performance testing tool used to test the stress and load handling capabilities of web applications and APIs. It facilitates the tester to simulate a large number of users and calculate the applications performance under different load conditions.SoapUI: An open-source testing tool for testing web services such as SOAP (Simple Object Access Protocol) and Restful APIs (Representational State Transfer)Read More: What are different software testing tools?Software Testing ModelsSoftware testing models provide structured approaches to ensure systematic and effective testing throughout the software development life cycle. Some commonly used testing models include:V-ModelThe V-Model is a development and testing approach that pairs each stage of the development process with corresponding testing activities to ensure thorough validation.Follows a sequential development process where each phase is directly linked to a specific testing phase.Emphasizes validation and verification at each stage, ensuring every part of the software is thoroughly tested.Testing activities, such as unit testing, integration testing, system testing, and acceptance testing, occur in parallel with the development stages.Ensures early detection of defects, reducing risks and ensuring higher quality throughout the development lifecycle.The Honeycomb ModelThis model emphasizes a multi-dimensional approach to software testing, ensuring thorough quality assurance across various aspects of software.Focuses on integration testing, especially for API-driven and cloud-based systems.Places less emphasis on unit testing and UI testing.Aims for comprehensive quality assurance by evaluating all aspects of the software.Emphasizes testing from multiple dimensions, including functionality, performance, security, usability, and more.Designed to create a well-rounded and reliable product by addressing all factors that affect user experience and software performance.Test Pyramid ModelThis model promotes a structured approach to testing, encouraging a larger number of low-level tests and fewer high-level tests to ensure efficient and scalable testing.Prioritizes unit tests at the base of the pyramid, followed by integration tests and fewer UI or end-to-end tests at the top.Focuses on testing individual components first, then their interactions, and finally, the entire system.Aims to catch defects early by having more tests at the lower levels, reducing the cost and complexity of fixing issues later.Encourages a scalable testing strategy that supports fast feedback and promotes high-quality software.Read More: What is Software Test Methodology? (With 6 Test Methodologies)Approach to Software TestingSoftware testing can be approached in two primary ways: manual and automated. Both approaches have advantages and use cases, depending on the nature of the project, the resources available, and the specific requirements of the testing process.Manual Software TestingManual testing involves testers executing test cases without the use of automation tools. This approach is often used for exploratory, usability, and ad-hoc testing where human insight is needed.Testers manually execute test cases to identify defects in the software.Ideal for situations where frequent changes are made to the software or when testing user interfaces.Useful for exploratory testing, usability testing, and one-time tests that are difficult to automate.Relies on human judgment to detect issues that may not be easily identified through automation.Time-consuming and repetitive for large-scale projects, making it less efficient for long-term testing.Read More: Top 15 Manual Testing ToolsAutomated Software TestingThe automation testing approach uses tools to execute test cases automatically, reducing the need for human intervention. It is well-suited for repetitive, time-consuming tasks and large-scale projects where speed and consistency are crucial.Involves writing scripts or using automation tools to execute predefined tests automatically.Saves time and effort for repeated tests, such as regression testing, over multiple iterations.Enhances efficiency and accuracy by eliminating human error in test execution.Ideal for large-scale projects or when frequent testing is needed.Requires initial investment in automation tools and script development, but it pays off over time with faster feedback and reduced manual effort.Read More: How to move from Manual to Automation TestingTypes of Software TestingSoftware testing is an indispensable part of Software Development Life Cycle (SDLC) as it is a process to assess the software and find all the possible bugs before releasing it into the market. Various types of testing which focuses on different aspects of the software needs to be performed to test any software thoroughly.Functional TestingFunctional testing verifies the software against requirement specification and validates that its functions and features work as expected. This testing is carried out by keeping the end users perspective in mind, like how an end user would use the software. Functional testing is a black box technique which means that the tester does not have access to the internal code or implementation and rather it should focus on only the system requirement.Unit Testing: Unit testing is a first level of functional testing where individual units of code are tested in isolation. This testing is often performed by the developers when the application is developed to ensure that the individual units of the software are working as expected. Developers use frameworks such as JUnit, TestNG to automate unit testing. This testing helps to identify bugs at the early stage thereby preventing it to become expensive to fix at the later stage.Integration Testing: Integration testing is the second level of functional testing where integration between different units/ components is tested. It checks whether the data flow between independent modules are working as expected or not. It is used to identify and resolve any issues surfaced while integrating one module with another. Integration testing can be performed via top-down, bottom-up, Big Bang and incremental approach.System Testing: When unit testing and integration testing is completed, system testing can be performed to test the entire software system as a whole to evaluate its compliance with required specifications. It is also called end to end testing as it assesses the interaction between different components and modules of the system to ensure that the end-to-end functionality works as expected.Read More: Differences Between Functional and Non-functional TestingNon-Functional TestingNon-functional testing assesses the aspects of software which are not related to its functionality but is related to its overall performance, usability, scalability, user experience and security. As functional testing emphasises testing the functionality of the software, non-functional testing focuses on how well the software performs.Security Testing: Security testing is a vital part of STLC which helps identify vulnerabilities, weaknesses and potential risks in a software system. This testing is used to ensure that the software is secure and protected against unauthorised access and severe attacks from the outside world.Performance Testing: Performance testing evaluates the softwares responsiveness, load capacity, stability, scalability and efficiency under several load conditions to ensure that it can handle the expected traffic.Usability Testing: Usability testing refers to testing the software from an end user perspective. It is used to evaluate the softwares user friendliness and overall experience to enhance the usability of software. It aims to identify usability issues, gather feedback and improve the user interface and experience.Compatibility Testing: Compatibility testing validates that the software application functions correctly across different environments, devices, operating systems, network and browsers. Given an example of a mobile application, in todays fast paced world, with so many devices with different screen-size, browsers, and operating systems, compatibility testing becomes an indispensable part in STLC. It is pivotal to test the software against all the environments and configurations before releasing it into the market.Read More: Types of Testing: Different Types of Software Testing in DetailFeatures of Software TestingSoftware testing possesses several features and characteristics that help in improving software quality, efficiency and reliability. Below are some of the key features:Early defect detection: The main objective of Software Testing is to find defects, bugs or errors in the software. Introducing Software testing from the beginning of the software development, helps in early defect detection which helps the development team to fix them on time and reduces the cost and effort of fixing defects at the later stage.Risk mitigation: Testing helps in identifying and mitigating risks associated with software failures. By encountering critical issues early, stakeholders can plan to manage project risks appropriately and effectively.Test Coverage: Comprehensive and detailed testing helps to ensure that all the important aspects of the software are thoroughly tested. It aims to achieve maximum coverage of the softwares functionality and requirements.Speed up the development process: Testing and development should go hand in hand means while the application is still developing, testing should happen in parallel to surface any potential bugs. This early detection of bugs helps the development team to fix them on time instead of fixing them at the later stage. This ultimately leads to speed up the development process and enhances it, as the development team need not wait till the end of the testing cycle for bug identification.Bug Free application: We know that a software can never be 100% bug free. However, continuous testing and retesting of bugs can turn out the software into its best form making it more usable and glitch-free.Quality Assurance: Software testing plays a vital role in quality assurance, as it ensures that the software meets the desired quality standards, is developed as per the system requirement and fulfils the user expectations.Verification and Validation: Verification and Validation is the process to investigate whether a software fulfils the specification, standards and the required purpose of the software. Verification (Are we building the product, right?) ensures that the software that is developed is right or not. Validation (Are we building the right product?) checks the validation of actual and expected outcome of the software.Documentation: Testing demands documenting every stage of STLC which includes Test Plan, Test Strategy, Test cases, Test results, Test closure and other testing artifacts. These documentations can serve as a reference which can be referred by any stakeholder at any given point of time.Customer Satisfaction: A happy customer is the key to a successful software. The main objective of any software delivering company is to cater the best features and experience to their customers through their product. Continuous testing, resolving the bugs, accepting and implementing customer feedback, all these contribute to enhancing the software productivity which ultimately helps in building the trust and satisfaction of the customer.Also Read: Software Testing Techniques: Explained with ExamplesBest Practices for efficient Software TestingA software can be successful only when it gains the confidence of the customers who actually use it. A poor-quality assurance can lead to a downfall in the number of customers. Software testing is essential to ensure the delivery of high-quality software, and therefore it should be periodically enhanced by following best practices, optimising team efforts, improving test coverage and early bug identification.Below are some of the best practices for efficient Software Testing:Test Planning: A comprehensive and formal test plan should be developed ahead of time to outline the testing scope, test resources, objectives, estimation, schedule and deliverables. A good documentation acts as a strong base for testing which can be revisited by any stakeholder to refer. Even if the testing team shuffles, or if anyone leaves or joins the team, documentation can help them understand the Software well.Also Read: Test Plan vs Test Strategy: Purpose & DifferencesShift-Left Testing: In a traditional testing approach (Waterfall model), typically testing used to start after the development was completed. However, in todays agile world of testing, the focus to test the system has been shifted to the left to initiate the testing from the start of the project. Shift Left Testing ensures that the bugs are caught early, which saves time and money and develops a healthy relation with the development team. This eventually accelerates the delivery of the product and aid in maximizing the test coverage.Read More: Why does DevOps recommend Shift Left PrinciplesTimely QA technical reviews: The purpose of the QA review is to ensure that the technical aspects of the software align with the established standards and quality requirements. It involves examining testing artifacts such as design documents, test cases, and other testing-related documents. These reviews are also called Formal technical reviews (FTR) which are mostly conducted on mature products by a set of stakeholders with different roles. Later an FTR report is created which addresses questions such as what was reviewed, who reviewed it, and what findings were discovered and the following decisions made. Some of the formal reviews are walkthrough and inspection.Test on real devices: Many simulators and emulators are available for testers to test their application across different devices. However, they lack to replicate the actual behaviour and performance of a real device. Hence, testing on real devices is essential as they provide real user conditions such as low battery, slow network, pop ups, network simulation and RAM conditions.Read More: Testing on Emulators vs Simulators vs Real DevicesEncourage communication and bug-triage: A developer and a tester can never be friends! Almost everyone has heard and experienced this very often. This nature cannot be completely fixed however this can be improved by having face-to-face communication, which can help both parties to express their views practically, avoid misunderstandings and come to a mutual conclusion to the problems faced during test execution.Bug triage is a meeting where the identified bugs are reviewed, prioritised and assigned to respective team members to work upon. This helps in prioritising the defects and ensuring that the most critical ones get addressed quickly.Run automated tests in CI/CD pipeline: To perform testing means to execute test cases, and over a period of time executing same test cases over and over again manually is time and resource consuming. Also, as manual testing is error prone, automated testing is preferable for such scenarios.Continuous Integration (CI) and Continuous Delivery (CD) aims to automate and streamline the process of building, testing and deploying. Automated testing should be integrated into the CI/CD pipeline where every code checked into the CI pipeline should be automatically tested. This will ensure that only bug-free code is deployed into the production system.Why choose BrowserStack?BrowserStack is a popular cloud-based testing platform that offers a comprehensive solution for testing web and mobile applications on real devices and browsers. Heres why you should choose BrowserStack for your software testing needs:Test on Real Devices and Browsers: Access 3500+ real devices and browsers for comprehensive testing across various OS.Automated Testing: Run tests at scale on BrowserStacks cloud, speeding up execution and reducing manual efforts.Visual Testing: Use Percy to detect design inconsistencies and UI bugs across screen sizes and browsers.Cross-Browser Compatibility: Ensure consistent performance across multiple browsers and OS for a seamless user experience.Seamless Integration: Integrates easily with popular testing frameworks, CI/CD tools, and project management systems for smooth workflows.Talk to an ExpertSoftware testing is key to ensuring quality and reliability. The Software Testing Life Cycle (STLC) offers a structured approach, while models like the Honeycomb, V-Model, and Test Pyramid guide effective validation. Both manual and automated testing are essential for evaluating functionality and performance.With the right tools and best practices, software testing leads to early defect detection and improved user satisfaction. BrowserStack provides access to real devices, automated testing, and seamless integration, helping teams deliver high-quality software.Test on Real Devices with BrowserStack Picture this. Your organization has been working hard on its product for two years. The release of the first version is approaching quickly, and everyone is excited. The stakes are high, so people are nervous, but theyre also confident on the quality of their work. Then, the big day finally comes and its terrible. Sure, the application looks great, and its features are fantastic. However, the app is riddled with embarrassing bugs. Users arent satisfied and the reviews are unforgiving. How could this be prevented? The answer is, of course, software testing. To err is human. No matter how much of a perfectionist you are, were all bound to make mistakes. Every organization has an end goal that comes with its own set of expectations. For some businesses, success is indicated by a high frequency of real outcomes matching expected results. But before reaching their end goal, every firm has to face the consequences of human errors. No enterprise can use manual error as an excuse for delivering a compromised product. To ensure a high-quality product, there has to be something in place to pick out errors. Software testing is an essential solution to this problem for software development companies. In this post, Im going to cover some software testing basics that you need to know.Here are some of the topics youll read about: What Do You Mean By Software Testing? The Importance of Software Testing What Are The Types of Software Testing? Functional Testing Non-Functional Testing Black-box Testing White-box Testing Gorilla Testing Grey-box Testing Software Testing Process How To Choose Between The Different Types of Software Testing? Enter The Test Automation Pyramid Lets get started. Fast and flexible authoring of AI-powered end-to-end tests built for scale. Start Testing Free Software testing is the process of finding errors in the developed product. It also checks whether the real outcomes can match expected results, as well as aids in the identification of defects, missing requirements, or gaps. Testing is the penultimate step before the launch of the product to the market. It includes examination, analysis, observation, and evaluation of different aspects of a product. Professional software testers use a combination of manual testing with automated tools. After conducting tests, the testers report the results to the development team. The end goal is to deliver a quality product to the customer, which is why software testing is so important. Its common for many startups to skip testing. They might say that their budget is the reason why they overlook such an important step. They think it would lead to no major consequences. But to make a strong and positive first impression, it needs to be top-notch. And for that, testing the product for bugs is a must. To really understand why software testing is important, we need to correlate it with real world examples, which has caused serious issues in the past, a few examples include: In October 2014, Flipkart an e-commerce in India company had an offer called the Big Billion Sale. When it was launched it had a lot of traffic and as a result, its website couldnt handle the enormous load of traffic leading to the website downtime, cancellation of orders etc. The reputation of the organization was badly impacted by this issue. In 2015, the Royal Bank of Scotland, due to a bug, couldnt process about 600,000 payments. Because of this, they were fined 66 million pounds Yahoo in September 2016, had a major data breach where 500 million users credentials got compromised. Recently, Okta, an American authentication firm, had a digital breach due to a software bug that may have affected their users details. This has also affected the reputation of Okta. Similarly, established organizations need to maintain their client base and their impression. So they have to ensure the delivery of flawless products to the end-user. Lets take a look at some points and see why software testing is vital to good software development. An enterprise can bring value to their customers only when the product delivered is ideal. And to achieve that, organizations have to make sure that users dont face any issues while using their product. The fool-proof way of doing it is by making your product bug-free. Organizations have to focus on testing applications and fix the bugs that testing reveals before releasing the product. When the team resolves issues before the product reaches the customer, the quality of the deliverable increases. When customers use the product, they are bound to reveal some sort of personal information. To prevent hackers from getting hold of this data, security testing is a must before the software is released. When an organization follows a proper testing process, it ensures a secure product that in turn makes customers feel safe while using the product. For instance, banking applications or e-commerce stores need payment information. If the developers dont fix security-related bugs, it can cause massive financial loss. The other part of security is not losing your data. Its common today for people to store data in cloud storage. You also probably have your photos and files stored on iCloud or Google drive. What if something goes wrong and you lose all your data? One of your nightmares isnt it? Security of a product not only protects information from hackers but also makes sure its not lost or gets corrupted. The days are gone when customers worked exclusively on hefty desktops. In the mobile-first age, testing a products device compatibility is a must. For instance, lets say your organization developed a website. The tester must check whether the website runs on different device resolutions. Additionally, it should also run on different browsers. Another reason why testing is gaining more importance is ever-increasing browser options. What works fine on Chrome may not run well on Safari or Internet Explorer. This gives rise to the need for cross-browser testing, which includes checking the compatibility of the application on different browsers. Software testing isnt a single thing. Instead, it comes in many different variations, which you can categorize according to several criteria. For instance, you can categorize testing types into manual or automated testing. When it comes to the automated variety, tests can be code-based or codelessand you can also have hybrid approaches that mix the best of both worlds. Tests can also be categorized regarding how much they know about the internal implementation of the system under test. Regarding this criterium, we can classify tests as white-box, black-box or grey-box. Finally, we can also group tests into functional and non-functional tests, depending on whether they validate the business requirements for the application. Functional testing verifies each function of an application or software. The tester verifies functionality with a specified set of requirements. So the source code of a software or an application doesnt play a major role in this case. Testing the behavior of the software is the main concern. The different types of functional testing include: Unit testing.In unit testing, the tester checks individual software components. The aim is to test whether the components behave according to the requirements. Integration testing. Integration testing deals with testing individual components or modules after they are combined in a group. System testing. Here, the tester executes test cases for verifying the compliance of integrated and completed software along with specifications. Sanity testing. This tests logical reasoning related to the working of the program. Smoke testing. Smoke testing tests simple and basic functionalities, such as if the user is able to log in or log out. Interface testing. These tests check whether the communication between two software systems is correctly carried out. Regression testing. This is probably one of the most important testing phases. Here, the old test cases of the entire application are executed after a new functionality has been implemented. Beta/acceptance testing. Here, the intended users try the product and report bugs. Non-functional testing considers parameters such as reliability, usability, and performance. A non-functional test might be checking how many users can log in to the system at the same time. Non-functional testing types include: Performance testing. The performance or speed of the application is tested under the required workload. Load testing. This tests an applications behavior under a huge workload. So, if youre testing a website, load testing checks the sites functionality and performance under high traffic. Stress testing. Stress testing determines software robustness by assessing whether its working beyond regular operation. Volume testing. This tests the performance of the system by loading the database to an increased volume of data. Security testing. Here, test cases are executed to check whether the system is safeguarded against sudden or deliberate attacks from internal and external sources. Compatibility testing. Test cases are executed to check whether the application is compatible with varying environments. For example, if youre testing a web application, compatibility testing deals with how the website works on different browsers or devices. Install testing. These tests check if a product works according to expectations after installation. Recovery testing. Here, testers determine an applications capacity to recover from hardware crashes and failures. Reliability testing. This procedure checks where an application can perform a particular task without failure within a specific timeframe. For example, suppose youre testing a cryptocurrency mining application. The scenario where the application can mine continuously for eight hours without crashing might be something you look for during reliability testing. Usability testing. Usability testing explores the end-users ease of use in terms of learning, operating, and preparing inputs and outputs. Compliance testing. This determines the systems compliance with external and internal standards. Localization testing. Here, testers check the behavior of a product according to local or cultural settings and environment. Based on the amount of information you know about the product to test it, software testing can be divided into different types: Black-box testing, White-box testing, and Grey-box testing. Black-box Testing In this type of testing, you have the least amount of information on how the product is built. You dont know about the structure of the product, its code, or logic. You would use the product as an end user would. Because in black-box testing, youd have the same amount of information as your customer, it is used for functional testing. This type of testing can only happen when the code is executed. Hence, dynamic testing is used. Dynamic testing is the type where you have to execute the code and test the product while the code execution is in process. It is mostly done to check how the would be when its up and running and how the user would experience it. White-box Testing In white-box testing, you have most of the information about the product. White-box testing is mostly used to make the code better. Finding inefficiencies in code, poor coding practices, unnecessary lines of code are identified in this type of testing. Most of the code optimization and security fixes happen as a result of this testing. White-box testing doesnt mainly focus on how the web application is working. It rather focuses on how it can be made better. You can make a lot of improvements to your product but the last few steps to make it perfect is difficult. And it cant be perfect until it has no issues whatsoever. Making it perfect requires a thorough inspection. Because a product in execution cant give you all the insights, youll have to check the code without execution. This is known as static testing. Static testing is also used in the early stages of development when its simple and you neednt wait for product deployment. Gorilla Testing Gorilla testing is a type of software testing, where a module is tested frequently using some random inputs, and ensuring that modules are checked with no errors. This type of testing is done manually and repeatedly, where only a few selected modules of the system are subjected to testing with the goal of determining whether or not the module is functioning properly. Other names for Gorilla testing include; torture testing, fault tolerance testing, and frustrating testing. Grey-box Testing In this type of testing, you have partial information about the product. This type of testing is helpful to find out bugs that the user wouldnt know about. To give you a very simple example, if youve designed an element to have a blue shade but it has a green shade. The user wouldnt know that its a bug because theyd think thats how its supposed to be. But your partial knowledge of the product would help you identify such bugs. Now that you understand what testing is all about, its time you know how to go about the software testing process. Software Testing Process Like any other process, software testing can also be divided into different phases. This sequence of phases is often known as the software testing life cycle. Lets look at them in brief. Planning Every process starts with planning. In this phase, you collect all the required details about the product. You collect a list of tasks that has to be tested first. If youre testing after a bug fix, then youd want to know what the bug was and whats the ideal behavior. Then you have to prioritize your checklist of tasks. If a complete team is involved, then division of tasks can also be done in this phase. Preparation Once you know what you have to do, you have to build the foundation for testing. This includes preparing the test environment, collecting test-cases, researching product features and test-cases. Gathering tools and techniques for testing and getting familiar with them should also be done here. Execution This is when you actually run tests on the product. You execute test-cases and collect the

results. Then you compare the results with the expected result and see if the product is working as expected or not. You make a note of all the successful and failed tests and test-cases. Reporting This is the last phase of software testing where you have to document all your findings and submit it to the concerned personnel. Test-case failures are of most interest here. A proper and clear explanation of tests run and outputs should be mentioned. For complex tests, steps to reproduce the error, screenshots, and whatever is helpful should be mentioned. As we know, in the current age of machines, everything that involves manual effort is slowly automated. And the same thing is happening in the testing domain. There are two different ways of performing software testingmanual and automation. Manual labor in any field requires a lot of time and effort. Manual testing is a process in which testers examine different features of an application. Here, the tester performs the process without using any tools or test scripts. Without using any automated tools, testers perform execution of different test cases. Finally, they generate a test report. Quality assurance analysts test the software under development for bugs. They do so by writing scenarios in an excel file or QA tool and testing each scenario manually. But in automated testing, testers use scripts for testing (thus automating the process). The pre-scripted tests run automatically to compare actual and expected outcomes. With test automation, when constant human intervention is not necessary, things like regression testing and execution of repetitive tasks dont seem like much effort. Now that youve got the gist of what manual and automated testing are, we need to clarify an important question. Is automation testing making manual testing obsolete? No. Even though the automatic performance of most processes takes place in automation testing, some manual labor is still a must. Generating the initial script for testing requires human efforts. Also, in any automated process, human supervision is mandatory. Automation simply makes the testing process easier. However, it doesnt make manual testing obsolete. You only get the best result by combining both manual and automated tests. Since testing is more efficient and speedy, theres a huge demand for automation testing compared to manual testing. And the reason is that it helps find more bugs in less time. By checking every single unit, automated testing also increases test coverage. As a result, an organizations productivity increases. As youve seen, software testing comes in many shapes and sizes. Each type provides a different kind of feedback, which means you cant use them interchangeably. Also, each type of testing comes with its own costs and associated challenges. Considering that your team and organization have limited resources, how can you choose between the many available types of testing in a way that maximizes test coverage, ensuring you can ship high quality software while using your resources in the most efficient way? Thats where the concept known as the test automation pyramid comes in handy. We do have an entire post about this concept, but heres the condensed version: the test automation pyramidaka the testing pyramid, for shortis a concept that helps you think about the different types of software testing and pick between them. At the bottom of the pyramid, you have unit tests.Unit tests are easier and cheaper to write than most other forms of testing. Since they dont talk to external dependencies, they run fast and are extremely precise in the feedback they provide. So, it makes sense to have lots of them. The middle of the pyramid is comprised of service tests, or integration tests. They offer a more realistic feedback than the unit tests, due to validating the integration between units and talking with real dependencies. But because of that theyre also slower to run, more complex to write and maintain, and offer a less precise feedback. Integration tests are valuable, but due to their limitations, it makes sense to employer fewer of them. Finally, the top of the pyramid contains end-to-end tests. End-to-end tests are the most realistic of all the software testing types since they exercise the application in the same way a real user would. However, they tend to be slower and fragile, besides being more expensive to write, maintain and execute. It pays to have these types of tests on your test suite, but youd be wise to have few of them. Testing Is Softwares Lifeline No company can underestimate the importance of delivering the best possible product to customers. And the types of testing keep evolving and the list keeps going on. Depending on the nature and scope of your products, you can run different testing procedures. Once the testing team gives the green signal, the deliverable is ready to go out into the market. But enterprises still need to keep in mind that customer trust doesnt come easily. To help earn customer trust, you need to provide consistent, reliable products. Thats why testing is an integral part of the software development life cycle. Arnab Roy Chowdhury wrote this post. Arnab is a UI developer by profession and a blogging enthusiast. He has strong expertise in the latest UI/UX trends, project methodologies, testing, and scripting. What to read next What Is End To End Testing? A Helpful Introductory Guide GUI Testing in Depth: Everything You Need to Understand

How to test fluency in reading. Fluency tester. How to test fluency in a language. How to test fluency.