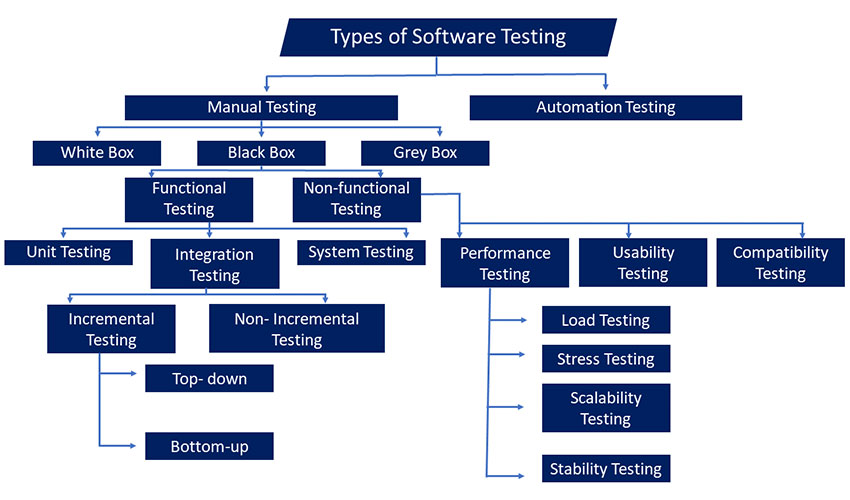
What are different types of Testing

The types of software testing are bifurcated into two main types i.e. Manual Testing and Automation Testing. Let’s understand both thoroughly.

**Manual Testing:**

It is one of the main types of software testing. It is a process in which the software gets tested manually to ensure the quality of the software for which exclusive test cases are written and tested for their pass or failure status. There are many types of manual testing & the most common are smoke testing, unit testing, functional testing and sanity testing.

**Classification of Manual Testing**

Manual testing is further classified into three types i.e. White Box Testing, Black Box Testing and Grey Box Testing. Let’s understand these three in brief.

**White Box Testing**

White box testing is a type of testing technique that examines the internal structure, code and logic of an application. It involves understanding the internal workings to design test cases and verify the accuracy of the software implementation.

**Black Box Testing**

The black box is a type of testing methodology where testers do not need to be able to decrypt or encrypt the software & they are not required to know the coding or internal structure of the software. The main focus of the black box testing method is to ensure operational or functional requirements.

**Types of Black Box Testing**

**Functional testing:**

Software testing tells us what the system does and is used to determine how each software application behaves according to the software requirements specification document. This type of QA test helps to check if the actual output follows the desired result & testers need not have to look at the source code of the application.

**Types of Functional Testing**

Now, that we understood what is functional testing, let’s understand the types of functional testing:

**Unit Testing**

It is a preliminary type of functional testing performed by engineers which focuses on a small software design called a unit. Tests were performed on one scalp or a group of related units to test their effectiveness and functionality. In Unit testing, a sample input is used and observed for the corresponding output view.

**Integration Testing**

Integration testing is a type of functional testing approach that evaluates the interaction and cooperation between different components or modules of an application. It focuses on verifying the integration of these components to ensure they function correctly together as a unified system.

**System Testing**

This type of testing is a QA test that incorporates several testing methods used to validate all software in terms of software, hardware, and networks. It usually evaluates a fully integrated software product and specializes in end-to-end system specifications.

**Non-functional testing**

It revolves around determining how well a system works under various conditions. It refers to various other forms of software testing & some of the most common testing methods are performance, load, stress, security and compatibility testing.

**Types of Non-Functional Testing**

Since we know there are types of testing, Non-functional testing is one of the types. In non-functional testing, it assesses software attributes beyond functionality. It covers performance, usability, security, reliability, and scalability. These tests uncover potential issues that impact quality and user experience. Here are common types of non-functional testing to ensure software effectiveness.

**Performance Testing**

It is one of the non-functional testing types that is used to test product readiness under the impact of heavy loads. It is done with conscientious testing procedures to determine the software’s responsiveness, reliability, and steadiness under varying user loads. It gets attained by measuring response times, output, and application usage levels under test.

**Types of Performance Testing**

Performance testing evaluates the responsiveness, scalability, stability and resource usage of software applications under various conditions. It helps identify bottlenecks and performance issues, ensuring optimal performance. Here are some common types of performance testing used to validate software performance.

**Load Testing**

It is a vital type of software testing methodology used to verify the ability of the system to manage the expected number of tasks. It also ensures system/application behaviour under normal and high load conditions.

**Stress Testing**

This test of software testing method executes to validate the behaviour of the system when the load rises above its design expectations.

**Scalability Testing**

Scalability testing assesses how well a software system can accommodate increasing workloads, user loads and data volumes while maintaining performance and efficiency. It helps determine if the system can scale up effectively as demand grows, ensuring a seamless user experience.

**Stability Testing**

Stability testing is a type of testing which evaluates the reliability and robustness of a software system under prolonged usage or stress conditions. It aims to identify any potential stability issues like memory leaks or crashes, ensuring the system can consistently perform without interruptions or failures over an extended period of time.

**Usability Testing**

Usability testing is a core test that can’t get disregarded as it determines whether a particular type of software testing system will be able to meet its purpose. The usability testing quickly determines the strengths and weaknesses of the system and assesses the overall usability of the product.

**Compatibility Testing**

Compatibility testing ensures that a software application functions as intended across different platforms, operating systems, browsers, devices and network environments. It verifies that the software is compatible with various configurations, allowing users to access and use the application seamlessly regardless of their chosen environment.

**Grey Box Testing**

Grey box testing is a mix of both black box and white box testing elements. Testers have partial knowledge of the internal workings of the software, allowing them to design test cases based on the system’s architecture and code structure while still focusing on the system’s external behaviour and functionality.

**Automated Testing**

It has governed the testing field as many open sources and paid automation testing tools are used to find the flaws and bugs. The software-automated testing process involves the creation of test scripts, the use of these test scripts and the entry of bugs in the software to control the crashes.

What are the Important Functional Testing Types?

Below is the list of some of the functional testing types:

**Smoke Testing**

This is yet another type of functional testing method performed by software testers. This type of test helps to ensure that the software or application under test is ready or stable for further testing. If the software is well constructed and fully functional, the testers accept the build and continue to perform the software testing process.

**Sanity Testing**

It is an effective testing methodology used to validate new additional tasks and get performed before detailed functional testing gets done. This testing method commences as soon as the build is approved to check whether the code changes introduced are working as expected.

**Regression Testing**

Regression testing involves retesting modified or updated software to ensure that existing functionalities work correctly and that no new issues or defects are introduced. It helps verify that changes in one part of the software do not negatively impact other areas, ensuring overall system stability and reliability.

**User Acceptance Testing**

User Acceptance Testing (UAT) involves testing a software system by end-users to ensure that it meets their specific requirements and functions as intended in their real-world environment. It serves as a final validation before deployment, allowing users to confirm if the software meets their needs and is ready for production use.

**Exploratory Testing**

Exploratory testing is a type of functional testing which has a dynamic approach where testers simultaneously learn, design and execute tests. It emphasizes creativity, intuition and the tester’s domain knowledge to uncover defects and understand system behaviour. Testers explore the software ad-hoc, adapting their testing based on immediate feedback and observations during the testing process.

**Adhoc Testing**

It is a type of testing which refers to a spontaneous and unplanned testing approach where testers execute test cases without following a predefined test plan. It involves exploring the software system randomly and informally to identify defects, issues, or unexpected behaviours that may not be covered by formal test cases. Ad hoc testing is often used to supplement other testing methods and uncover unforeseen problems.

**Security Testing**

Security testing is a type of software testing which is conducted to assess the robustness of a software system against potential threats and vulnerabilities. It involves evaluating the system’s ability to protect data, prevent unauthorized access, ensure compliance with security standards and detect and mitigate security risks thus safeguarding the integrity and confidentiality of the system.

**Globalization Testing**

Globalization testing evaluates the software’s ability to function correctly and effectively across different regions, languages and cultures. It verifies if the application can handle diverse date and time formats, character encodings, currency symbols and language translations, ensuring its compatibility and usability in a global context.

What Are The Key Non-functional Software Testing Types?

**Performance Testing**

It is a method of non-functional testing type and is used to test product readiness under the impact of heavy loads. It is done with conscientious testing procedures to determine the software’s responsiveness, reliability, and steadiness under varying user loads. It gets attained by measuring response times, output, and application usage levels under test.

**Load Testing**

Load testing is a type of non-functional testing that measures the software’s performance under specific anticipated or simulated user loads. It assesses the system’s ability to handle the expected concurrent user activity, data volumes, and transaction rates, identifying performance bottlenecks, response times, and resource utilization to ensure optimal system scalability and stability.

**Stress Testing**

Stress testing is a non-functional testing type that evaluates the software’s behaviour and performance under extreme conditions beyond its normal operational capacity. It involves subjecting the system to high user loads, excessive data volumes, or limited system resources to identify any weaknesses, bottlenecks, or failures. The goal is to determine the software’s stability, responsiveness, and recovery capabilities, ensuring it can handle unexpected or peak loads without crashing or compromising its functionality. By pushing the system to its limits, stress testing helps uncover potential issues and allows for necessary optimizations and improvements to enhance the software’s overall performance and reliability.

**Usability Testing**

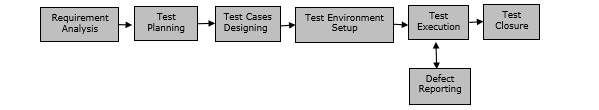
Usability testing is a core test that can’t get disregarded as it determines whether a particular software system will be able to meet its purpose. The usability testing quickly determines the strengths and weaknesses of the system and assesses the overall usability of the product.

**2. What are the different SDLC phases**

**STLC Phases**

STLC has the following different phases but it is not mandatory to follow all phases. Phases are dependent on the nature of the software or the product, time and resources allocated for the testing and the model of SDLC that is to be followed.

**STLC Phases**



**Requirement Analysis −** When the SRD is ready and shared with the stakeholders, the testing team starts high level analysis concerning the AUT (Application under Test).

**Test Planning −** Test Team plans the strategy and approach**.**

**Test Case Designing −** Develop the test cases based on scope and criteria’s.

**Test Environment Setup −** When integrated environment is ready to validate the product.

**Test Execution −** Real-time validation of product and finding bugs.

**Test Closure −** Once testing is completed, matrix, reports, results are documented.

**3. As manual tester what are the qualities you possess**

After all, after a crucial part in the SDLC. Other essential qualities for a QA manual tester include excellent analytical skills and the ability to communicate effectively with others (written and verbal). You should also demonstrate an ability to think creatively and problem-solve. Manual testers should have many qualities, including:

**Communication skills**

Good written, speaking, listening, and reading skills are important. Manual testers should also be able to convey complex ideas clearly.

**Analytical skills**

Manual testers should be able to identify and isolate defects in complex systems. They should be able to approach a problem logically, break it down into smaller components, and systematically examine each part to find the root cause of the problem.

**Attention to detail**

Manual testers should have strong attention to detail.

**Problem-solving abilities**

Manual testers should have excellent analytical and problem-solving abilities.

**Computer programming knowledge**

Manual testers should have a basic understanding of the functionality of the language used. They should also have a solid knowledge of database and programming fundamentals.

**Understanding of software testing methodologies and techniques**

Manual testers should have a thorough understanding of software testing methodologies and techniques

**4. What is the difference between Waterfall and Agile methodologies in SDLC?**

Agile and Waterfall are two popular software development life cycle (SDLC) methods used to plan, execute, and deliver software projects**.**

Agile is an iterative, incremental, and adaptive approach to software development. Agile teams work in short sprints, typically two to four weeks long, and focus on delivering small, usable pieces of software at the end of each sprint. Agile methodologies, such as Scrum and Kanban, prioritize flexibility, collaboration and customer satisfaction, and are well suited for projects with changing requirements and uncertainty.

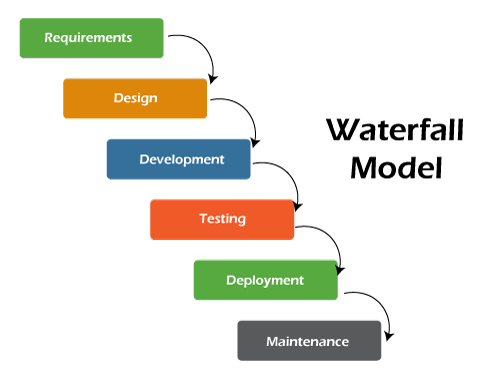
Waterfall, on the other hand, is a more traditional, linear approach to software development. In Waterfall, development is divided into distinct phases such as requirements gathering, design, development, testing, and deployment. Each phase must be completed before the next one can begin. Waterfall is known for its well-defined process and is often used in projects where requirements are well-understood and unlikely to change.



Agile development methodology and testing practices have worked wonders for several organizations with positive aspects. The positive aspects of agile are not hidden. They are very much visible in organizations. There are some of the important points related to the agile model listed as follows -

* Agile focuses on customer feedback, collaboration, small and rapid releases.
* Its purpose is to manage complex projects.
* The Agile produces better application suites with the desired requirements. Moreover, it can quickly adapt according to the changes made on time during the project life.
* It has a small team size. Therefore, fewer people work on it so that they can move faster.
* The agile model is not a suitable model for small projects. The expenses of developing the small projects using agile are more than compared to other models.
* In agile methodology, the interaction of customers is very high, as after each iteration an incremental model is deployed to customers.

Now, let's move forward to the next model, i.e., the waterfall model.



It is one of the easiest and traditional model to manage. Because of its traditional development nature, each phase has specific deliverables and a review process. The waterfall model works well in smaller size projects where requirements are easily understandable.

The waterfall model is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases. The development in the waterfall model is seen as flowing steadily downwards (like a waterfall) as it is a continuous software development model. This model is named "Waterfall Model", because its diagrammatic representation resembles a cascade of waterfalls. Some important points related to the waterfall model are listed as follows -

Waterfall model is not an ideal model to develop a large scale project size.

* The requirements in the waterfall model should be clear cut at the beginning time; otherwise, it may lead to a less effective method.
* In the waterfall model, it is hard to move back in order to make changes in the previous phase.
* The testing process in the waterfall model starts after the completion of development. So, there is a high chance of bugs to be found later in the project development.

