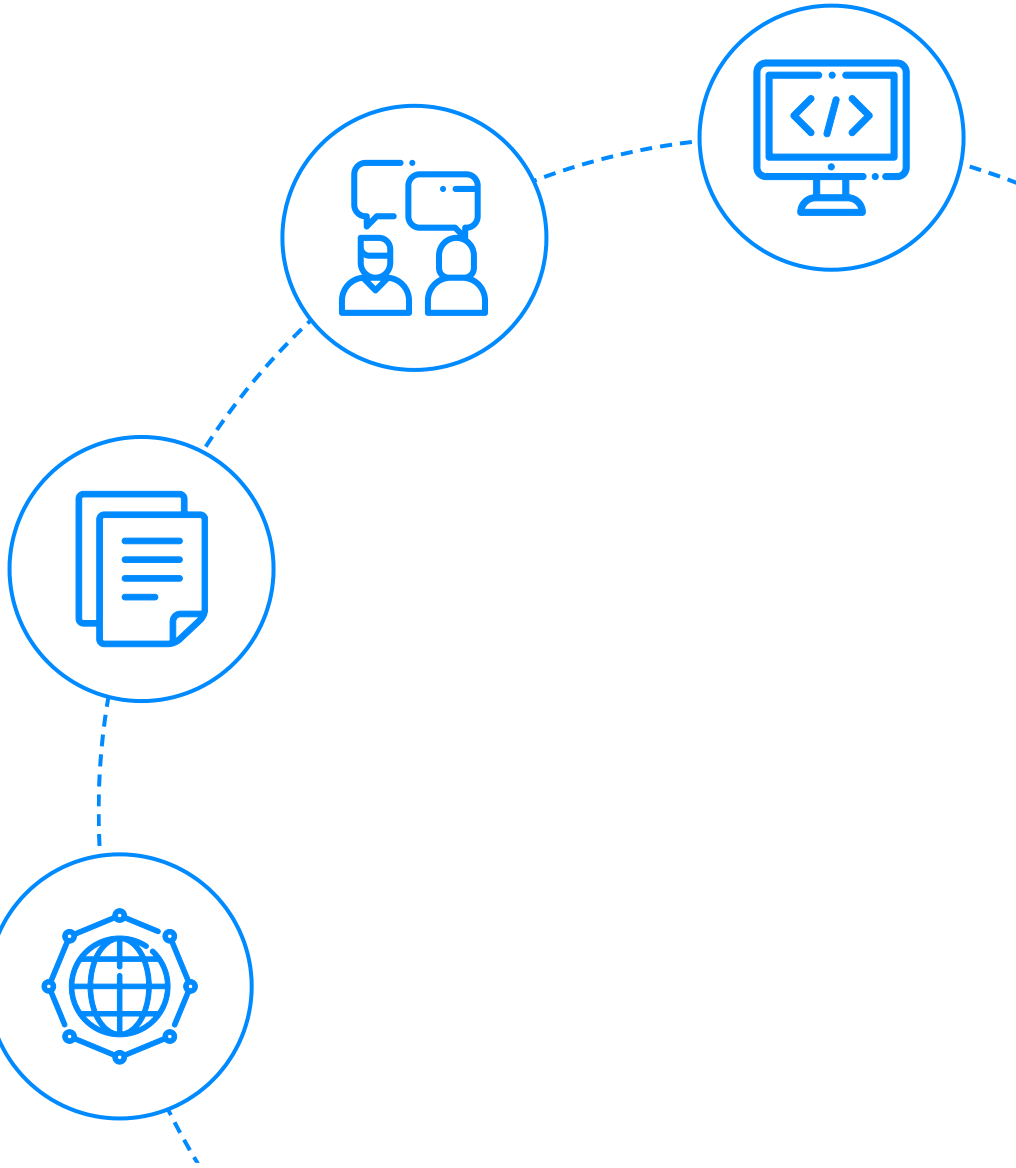
InterviewBit

QA Interview Questions



To view the live version of the page, click here.

Contents

QA Interview Questions for Freshers

QA Interview Questions for Experienced

QA Interview Questions for Experienced (.....Continued)

19. What do you understand about severity and priority of a defect in the context of quality assurance? Differentiate between them.
20. What do you mean by quality audit in the context of quality assurance?
21. How do you figure out how much testing each piece of software requires in the context of Quality Assurance?
22. Differentiate between load testing and stress testing.
23. Differentiate between functional and non functional testing.
24. What do you understand by black box and white box testing? Differentiate between them.
25. What do you understand about bug/ defect triage in the context of quality assurance?
26. What do you understand about stubs and drivers? Differentiate between them.

Let's get Started

What is Quality Assurance (QA)?

Quality Assurance comprises two terms quality and assurance. So, before we understand what quality assurance is, let us see what the terms quality and assurance mean. The term "quality" refers to a product or service that is "fit for use and purpose."

Assurance is nothing more than a positive statement about a product or service that inspires trust. It is the assurance that a product or service will perform successfully. It ensures that the product will perform flawlessly and in accordance with the expectations or criteria.

Starbucks lost millions of dollars in sales in 2015 due to a problem with their daily system refresh, which resulted in the closure of point-of-sale registers in many locations across the United States and Canada. Until the systems were rebuilt and restored, coffee businesses were compelled to give away free drinks. This clearly depicts how important it is for the system to be superior in quality.

In software testing, quality assurance is described as a technique for ensuring the quality of software products or services is given to clients by a company. Any systematic process of verifying whether a product or service fulfils stated requirements is known as quality assurance (QA). Quality assurance is concerned with making the software development process more efficient and effective in accordance with the quality standards established for software products. QA Testing is a popular term for Quality Assurance. QA is a proactive procedure that identifies strategies to avoid bugs in the software development process. Stakeholders, Business Analysts, Developers, and Testers are all required to participate in the quality assurance (QA) phase of the software development lifecycle (SDLC). By describing and establishing the requirements for both the software development process and quality standards, the QA process aids the project team's productivity. A quality assurance system is designed to boost consumer trust and credibility while also enhancing work procedures and efficiency, allowing a company to compete more effectively.

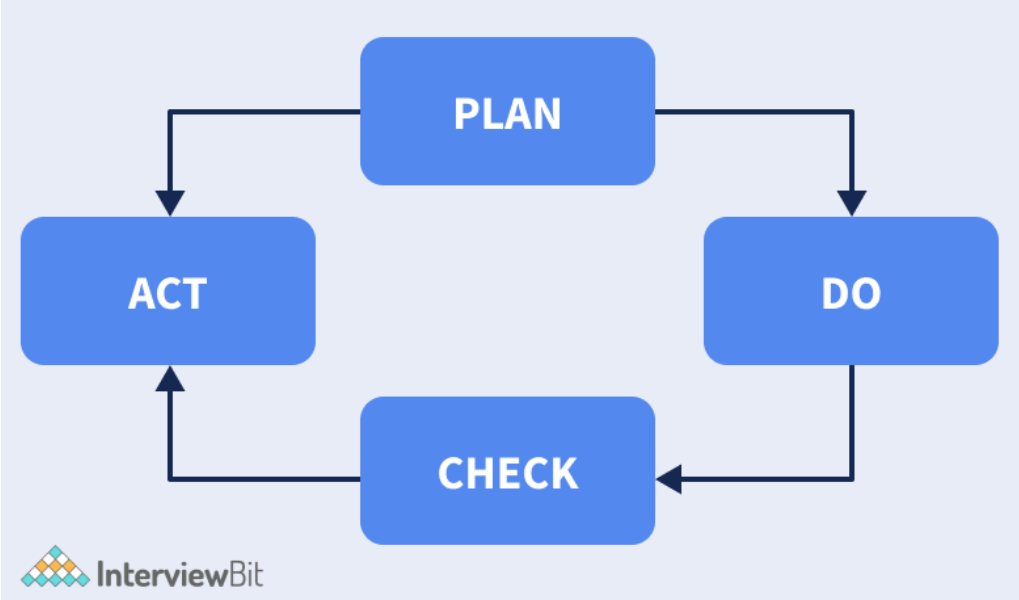
The ISO (International Organization for Standardization) is a driving force behind quality assurance methods and the mapping of quality assurance processes. QA is frequently used in conjunction with the ISO 9000 international standard.

To help you prepare for your forthcoming interview, we've covered the most relevant QA Testing interview questions for freshers as well as for experienced **QA Engineers**.

QA Interview Questions for Freshers

1. What is the lifecycle of a Quality Assurance Process?

Every process in Quality Assurance involves the PDCA (**Plan Do Check Act)** cycle, often known as the **Deming cycle**. The following are the phases of this cycle:



- **Plan:** The organisation should plan and develop process-related objectives, as well as the methods necessary to provide a high-quality final product. Here, Quality Assurance ensures that the planning made takes into consideration the quality of the product.
- **Do:** This phase involves process development and testing, as well as "doing" changes to processes. Here, Quality Assurance ensures that the processes followed during development maintain the quality of the product.
- **Check:** In this phase, processes are monitored, modified, and checked to see if they achieve the intended goals. Here, Quality Assurance ensures that the processes are checked thoroughly so that no defects might be missed.
- **Act:** In this phase, a Quality Assurance tester should take the steps that are required to improve the processes.

These stages are done to guarantee that the organization's procedures are assessed and improved on a regular basis.

2. Differentiate between Quality Assurance and Testing.

Software Testing: Software testing is a method of investigating a system to see how it works according to product requirements and customer expectations and identify potential flaws. To test the product, find bugs, and see if they've been fixed, a variety of methods are utilised. Customers can use testing to check if the generated product satisfies their expectations in terms of design, compatibility, and functionality, among other things.

Validating the product against specifications and client requirements, as well as detecting and reporting flaws, are all part of the testing process. To detect software flaws, it uses a variety of testing methodologies such as functional, non-functional, and acceptability testing. Furthermore, before the product is released to the client, the purpose of software testing is to ensure that any found faults are entirely corrected with no side effects.

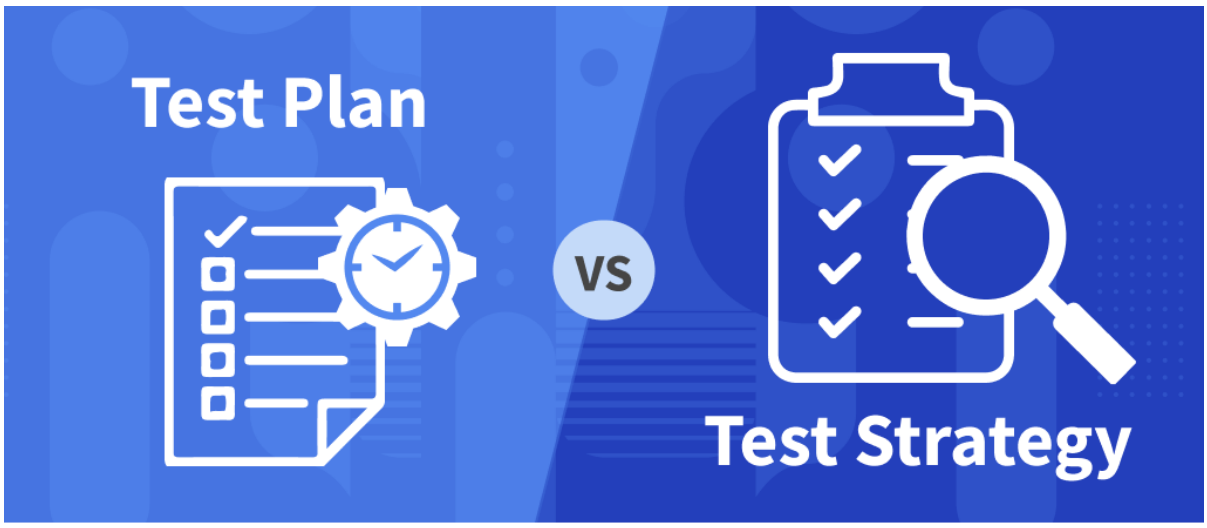
The following table lists the differences between Quality Assurance and Testing:

Quality Assurance	Testing
It is a subset of the Software Development Lifecycle (SDLC).	It is a subset of Quality Control (QC).
It is process-oriented.	It is product-oriented.
It is preventive in nature (tries to prevent defects from being present in the product).	It is corrective in nature (tries to correct defects present in the product).
Quality Assurance is done to prevent defects in the product.	Testing is done to find and fix defects in the product.
Quality Assurance ensures that the processes and procedures followed by the team are in place.	Testing confirms that the product conforms to the specifications required.
In Quality Assurance, the focus is on the processes that operate on the product.	In Testing, the focus is on the end product itself.
It is a proactive process (a proactive strategy focuses on preventing problems from arising).	It is a reactive process (a reactive approach focuses on reacting to events after they have occurred).
Quality Assurance needs to be implemented by the whole team.	In Testing, only the testing team is concerned.

3. Differentiate between Test Plan and Test Strategy.

- **Test Plan:** A test plan is a document that illustrates the test procedure, destinations, timetable, estimation, and expectations, as well as the assets needed for testing. It motivates us to determine the effort required to approve the kind of application being tested. The test plan serves as a diagram to guide software testing exercises as a well-defined method that the test manager closely monitors and controls. Test plan id, highlights to be tested, test systems, testing assignments, highlights pass or bomb criteria, test expectations, duties, and timeline, and so on are all included in the test plan.
- **Test Strategy:** In software testing, a test strategy is a collection of guiding principles that specifies the test design and control how the software testing process is carried out. The Test Strategy's goal is to give a systematic method to software testing so that quality, traceability, reliability, and better planning may be assured.

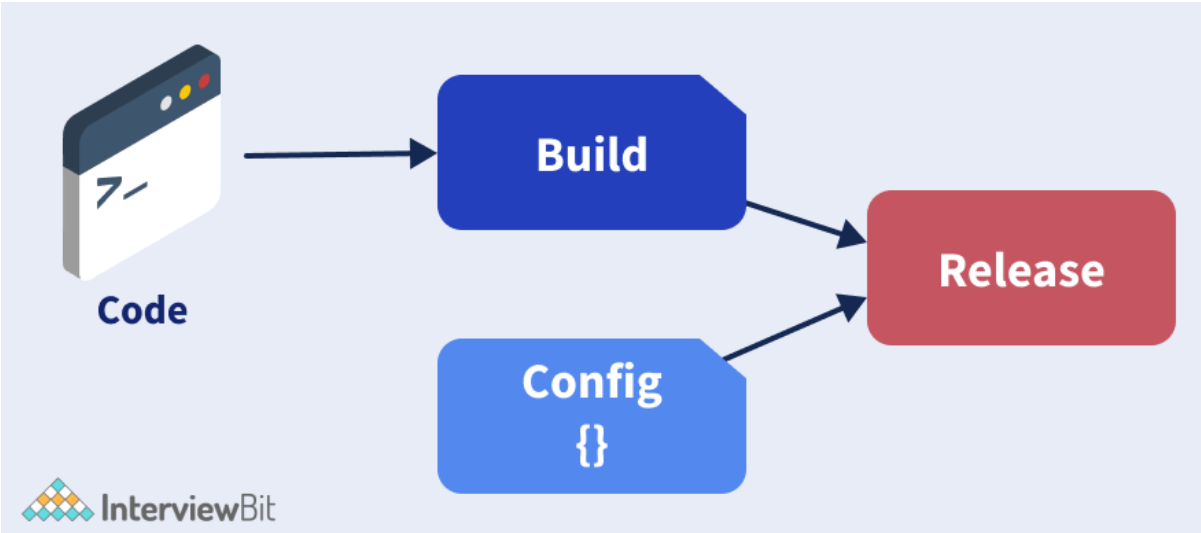
The following table lists the differences between [Test Plan and Test Strategy]():



Test Plan	Test Strategy
A software project test plan is a document that specifies the scope, purpose, approach, and emphasis of a software testing process.	A test strategy is a set of rules that describes how to develop tests and outlines how they should be carried out.
Test plan elements include the following: • Test plan id • Features to be tested • Test procedures • Testing tasks • Features pass or fail criteria • Test deliverables • Responsibilities • Schedule, among others.	Following are the components of a test strategy : • Objectives and scope • Documentation formats • Test methodologies • Team reporting structure • Client communication strategy
The specifications of the testing process are described in the test plan.	The general approaches are described in the test strategy.
A testing manager or lead executes a test plan that specifies how to test when to test, who will test, and what to test.	The project manager implements a test strategy. It specifies which approach to use and which module to test.
Changes to the test plan are possible once it has been created.	It is impossible to alter the test strategy once it has been created.

4. What do you mean by build and release in the context of quality assurance? Differentiate between them.

- **Build:** A build is a software or application that is ready to be tested. Developers create software, which they then hand over to testers to test. It's a broad phrase that refers to any application that will be examined. Developers can either create a complete program or add a new feature to an existing one. Then that program, together with the new functionality, is dubbed BUILD, and it is put through its tests by testers.
- **Release:** After development and testing, the release is the finalized application. The testing team verifies the program and releases it to the customer after testing it. It's possible that a single release will have many builds. As a result, it is the program that is supplied to the customer when the development and testing phases are completed. Furthermore, the release is based on builds, and there may be multiple builds.



The following table lists the differences between build and release:

Build	Release
Build refers to a version of the software or application which the development team hands over to the testing team.	Release refers to the software which the testing team hands over to the end customers.
The build version of the software requires testing to be done on it. Generally, sanity testing is performed on a build version.	The release version of software no longer requires testing to be done on it.
The build version of the software is made more frequently than the release version.	Release versions of the software are made less frequently than the build version.

5. What do you understand about bug leakage and bug release?

- **Bug Leakage:** When a bug is discovered by an end-user, that should have been caught in earlier builds/versions of the application, a bug leakage occurs. Bug leaking refers to a fault that exists during testing but is not discovered by the tester and is subsequently discovered by the end-user.
- **Bug Release:** When a particular version of the software is released with a collection of known bugs/defects, it is referred to as a bug release (s). Bugs of this type are frequently of low severity and/or priority. When the company can afford the existence of a bug in the released software rather than the time/cost of repairing it in that version, this is done. In most cases, these bugs are disclosed in the Release Notes.

6. What do you understand about Traceability Matrix (TM) in the context of quality assurance?

A Traceability Matrix is a document that connects any two baseline documents that require a many-to-many link to ensure that the relationship is complete. It's used to keep track of requirements and make sure they're being met on the present project.

The following are the three major components of a traceability matrix:

- **Forward Traceability:** This matrix is used to determine whether the project is progressing in the appropriate direction and for the correct product. It ensures that each specification is applied to the product and that each specification is adequately tested. It connects test cases to requirements.
- **Backward or reverse Traceability:** It is used to verify that the present product is still on track. The goal of this form of traceability is to ensure that we are not expanding the project's scope by adding code, design features, testing, or other activities not stated in the requirements. It connects requirements to test cases.
- **Bidirectional Traceability:** This traceability matrix ensures that all criteria are covered by test cases in both directions (forward and backward). It examines the impact of a change in requirements caused by a work product defect, and vice versa.

7. What do you understand about defect leakage ratio in the context of quality assurance?

Software testers utilise defect leakage as a metric to determine the effectiveness of Quality Assurance (QA) testing. It's the ratio of the total number of flaws attributed to a stage (which are captured in subsequent stages) to the sum of the total number of defects captured in that stage and the total number of defects assigned to a stage (which are captured in subsequent stages). Defect leakage is a metric that counts the percentage of faults that leak from one testing stage to the next, as well as demonstrating the effectiveness of software testers' testing. The testing team's worth, on the other hand, is only confirmed when defect leaking is small or non-existent.

8. Differentiate between Quality Assurance (QA) and Quality Control (QC).

Quality Control (QC): A systematic set of techniques used to assure the quality of software products or services is known as quality control in software testing. By testing and reviewing the software product's functional and non-functional requirements, the quality control process ensures that it fulfils the actual needs.

The following table lists the differences between Quality Assurance (QA) and Quality Control (QC):

Quality Assurance	Quality Control
It is a method that focuses on assuring that the specified quality will be met.	It is a method that focuses on achieving the desired level of quality.
The goal of quality assurance is to avoid defects.	The goal of quality control is to find and correct flaws.
It's a way to keep track of quality. (Verification)	It's a method for verifying the quality. (Validation)
It does not entail running the software.	It always entails running software.
It is a Proactive measure (method of prevention).	It is a Reactive measure (a remedial technique).
It is the method for producing deliverables.	It is the technique for ensuring that deliverables are correct.
QA is involved in the entire software development process.	QC is involved in the entire software testing process.
Quality Assurance establishes standards and processes in order to achieve client expectations.	While working on the product, Quality Control ensures that the standards are followed.
It is carried out prior to Quality Control.	It is only carried only once the QA activity has been completed.

9. What do you mean by monkey testing in the context of quality assurance?

Monkey testing is a software testing technique in which the tester inserts any random inputs into the software application without using predefined test cases and observes the software program's behaviour to see if it crashes. The goal of monkey testing is to use experimental ways to uncover faults and problems in software applications. Monkey testing is a sort of black-box testing that involves supplying random inputs to a system in order to check its behaviour, such as whether it is crashing or not. Monkey testing does not necessitate the creation of test cases. It can also be automated, in the sense that we can develop programs or scripts to produce random inputs in order to test the system's behaviour. When undertaking stress or load testing, this technique comes in handy.

Monkeys are divided into two categories:

- **Smart Monkeys:** The smart monkeys are those who have a basic understanding of the application. They know which of an application's pages will redirect to which page. They also know whether the inputs they're giving are valid or not. If they discover an error, they are wise enough to report it as a bug. They are also aware of the menus and buttons.
- **Dumb Monkeys:** Dumb Monkeys are individuals who are completely unaware of the application. They have no idea where an application's pages will reroute. They give random inputs and are unaware of the application's beginning and finish points. Despite the fact that they have no knowledge of the application, they discover bugs such as environmental failure or hardware failure. They also have limited knowledge of an application's functioning and user interface.

10. What do you mean by gorilla testing in the context of quality assurance?

Gorilla Testing:

Gorilla testing is a method of software testing in which a module is frequently tested based on some random inputs, ensuring that the module's operations are checked and that there are no problems in the module. So it's also known as Torture Testing, Fault Tolerance Testing, or Frustrating Testing because of the Gorilla Testing pattern. It's a manual test that's done over and over again. In Gorilla Testing, testers and developers work together to regularly evaluate a module's functionality.

11. Differentiate between gorilla testing and monkey testing.

The following table lists the differences between gorilla testing and monkey testing:

Gorilla Testing	Monkey Testing
Gorilla testing is a method of software testing in which a module is frequently tested based on some random inputs, ensuring that the module's operations are checked and that there are no problems in the module.	Monkey testing is a method of software testing that evaluates the behaviour of the system and validates whether it crashes or not based on some random inputs and no test cases.
The primary goal of Gorilla testing is to determine whether or not a module is functioning properly.	The primary goal of monkey testing is to determine whether or not a system will crash.
Gorilla testing is a type of manual testing that is done repeatedly.	Monkey testing is a sort of random testing that does not involve test cases.
Only a few select modules of the system are subjected to this testing.	This testing is carried out over the entire system.
Unit testing primarily employs the Gorilla Testing method.	In System Testing, the Monkey Testing approach is primarily employed.
Torture Testing, Fault Tolerance Testing, and Frustrating Testing are all terms used to describe gorilla testing.	Random testing, Fuzz testing, and Stochastic testing are all terms used to describe monkey testing.

12. Explain what is a testware in the context of quality assurance.

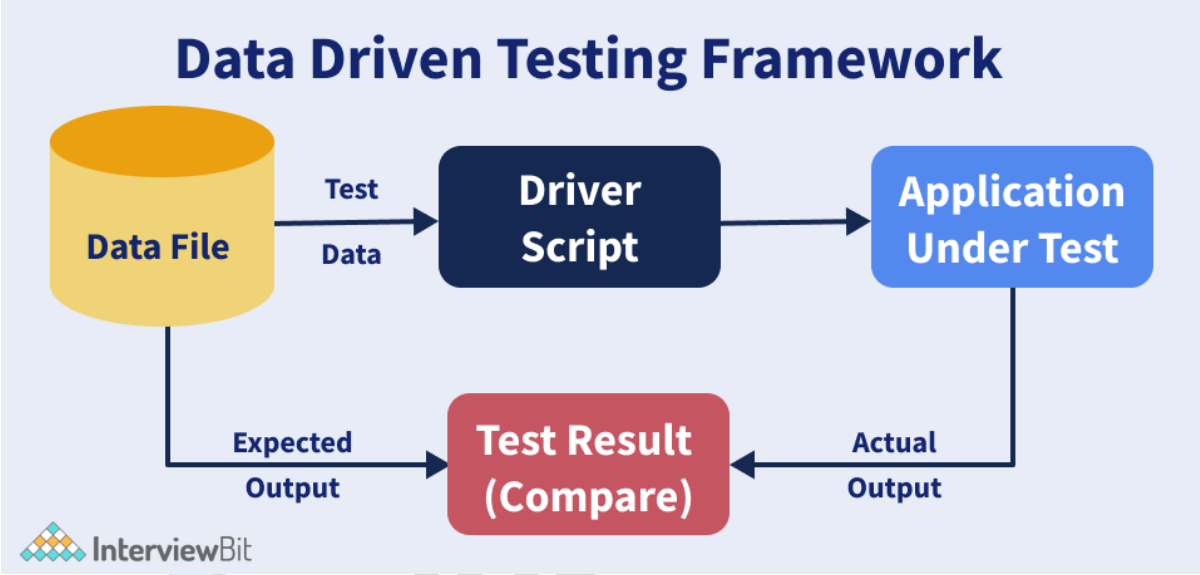
Testware is a collection of software created for the specific purpose of software testing, particularly software testing automation.

For example, automation testware is created to run on automation frameworks. All utilities and application software that work together to test a software package but do not necessarily contribute to operational purposes are referred to as testware. As a result, testware is only a working environment for application software or portions thereof, rather than a static configuration. It contains artifacts created during the testing process that is needed to plan, develop, and execute tests, such as documentation, scripts, inputs, expected outcomes, set-up and clear-up processes, files, databases, environment, and any other software or tools used during testing. Both verification and validation testing methodologies are used to create testware. Testware, like software, consists of codes and binaries, as well as test cases, test plans, and test reports. Testware should be preserved and faithfully maintained under the direction of a configuration management system.

13. What do you understand about data driven testing?

Data-Driven Testing is a software testing technique that stores test data in a table or spreadsheet format. Testers can use data-driven testing to enter a single test script that can run tests for all test data from a table and expect the test results to be returned in the same table. It's also known as parameterized testing or table-driven testing.

Because testers usually have several data sets for a single test, Data-Driven Testing is critical. Creating different tests for each data set can be time-consuming. Data-driven testing allows data to be kept separate from test scripts, and the same test scripts can be run for multiple combinations of input test data, resulting in more efficient test results.



The above image depicts the process of data-driven testing. Test data is taken from a data file and tested on the application and then the produced output is compared with the actual output.

14. How would you ensure that your testing is thorough and comprehensive?

The Requirement Traceability Matrix and Test Coverage Matrix will assist us in determining whether or not our test cases are adequately covered. The requirement traceability matrix will assist us in determining whether the test conditions are sufficient to fulfil all of the requirements. Coverage matrices will assist us in determining whether the test cases are sufficient to satisfy all of the Requirement Traceability Matrix test conditions.

The below image shows a sample Requirement Traceability Matrix:

Requirements	Test Conditions						
	TC_01	TC_02	__	__	__	__	TC_175
Req_01							
Req_02							
Req_03							
,,							
,,							
,,							
,,							
Req_100							

InterviewBit

The below image shows a sample Test Coverage Matrix:

Test Conditions	Test Cases						
	TC_01	TC_02	__	__	__	__	TC_175
TestCon_01							
TestCon_02							
TestCon_03							
,,							
,,							
,,							
,,							
Req_100							

InterviewBit

15. What are the various artifacts to which you refer when writing test cases?

Following are the various artifacts that we can refer to while writing test cases :

- Specification of functional requirements.
- Document that explains what the requirements are.
- Wireframes.
- Use Cases.
- User Stories.
- Acceptance criteria.
- User Acceptance Testing Test cases.

16. What do you mean by a test case? What are some good practices for writing test cases?

A test case is a collection of actions performed to ensure that a specific feature or operation of your software program is working properly. A Test Case is a set of test procedures, data, preconditions, and postconditions created for a specific test scenario in order to verify any requirement. The test case contains specified variables or conditions that a testing engineer might use to compare expected and actual outcomes in order to assess whether a software product meets the customer's needs.

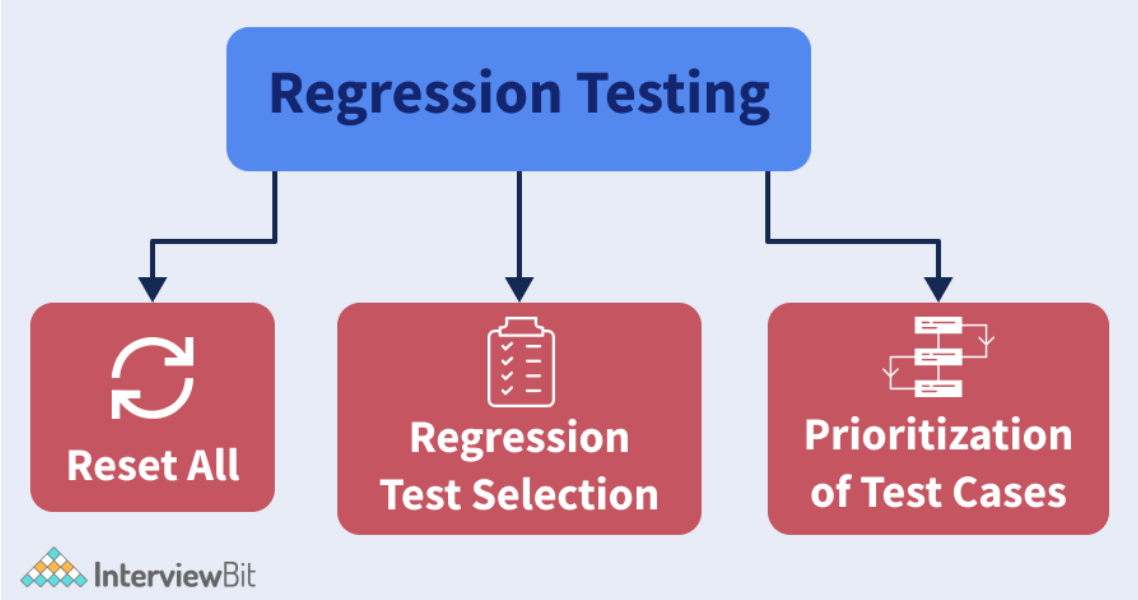
Following are some good practices for writing test cases :

- **Simple and transparent test cases are required:** Make your test cases as simple as possible. They must be clear and straightforward because the test case author may not be able to perform them. Use declarative language such as "go to the main page," "input data," "click here," and so on. This makes it easier to understand the test stages and speeds up the testing process.
- **Create a test case that considers the end-user:** Any software project's ultimate goal is to produce test cases that fulfil client requirements and are simple to use and run. A tester must write test cases from the standpoint of the end-user.
- **Repetition of test cases should be avoided:** Test instances should not be repeated. If a test case is required for the execution of another test case, use the test case id in the preconditioned column to refer to it.
- **Make certain you have complete coverage:** Make sure you write test cases to ensure you've covered all of the software requirements in the specification document. To verify that no functions or conditions are left untested, use the Traceability Matrix.
- **Don't Make Assumptions:** When creating a test case, don't make assumptions about the functioning and features of your software application. Follow the specifications in the specification documents.
- **Self-cleaning:** The test case you write must restore the Test Environment to its previous condition and should not render it unusable. This is particularly true when it comes to configuration testing.

QA Interview Questions for Experienced

17. What do you understand about regression testing? Which test cases should be selected for regression testing?

Regression Testing is a sort of software testing used to ensure that a recent program or code modification hasn't broken existing functionality. Regression Testing is just a full or partial re-execution of previously executed test cases to confirm that current functionality is working properly. This testing ensures that new code modifications do not have unintended consequences for current functionality. It ensures that the old code continues to function after the most recent code modifications have been made.



According to industry data, a large proportion of defects reported by customers were caused by last-minute bug patches that had unintended consequences, making selecting the Test Case for regression testing an art and not an easy task. The following test scenarios can be used to create effective regression tests :

- Test scenarios with a high number of flaws
- Test cases covering features that are more visible to the users
- Test cases that test the product's fundamental features
- Test cases of functionalities that have experienced significant and recent alterations
- All Test Cases for Integration
- All Complex Test Cases
- Cases of boundary value tests.

18. Explain risk in the context of quality assurance. What are the five dimensions of risk?

When it comes to software testing, Risks are potential issues that could jeopardise the project stakeholders' goals. It's the likelihood of a bad or unfavourable result. A risk is something that has not yet occurred and may never occur; it is a potential concern. The likelihood of a risk becoming an outcome is proportional to the degree of risk linked with the potential for negative consequences.

Most people, for example, are expected to have a cold at some point in their lives, frequently multiple times. A healthy person, on the other hand, is spared any major effects. As a result, this individual's overall risk of catching a cold is low. On the other hand, for an elderly person with breathing issues, the risk of catching a cold is significant. As a result, the overall chance of catching a cold in his instance is significant.

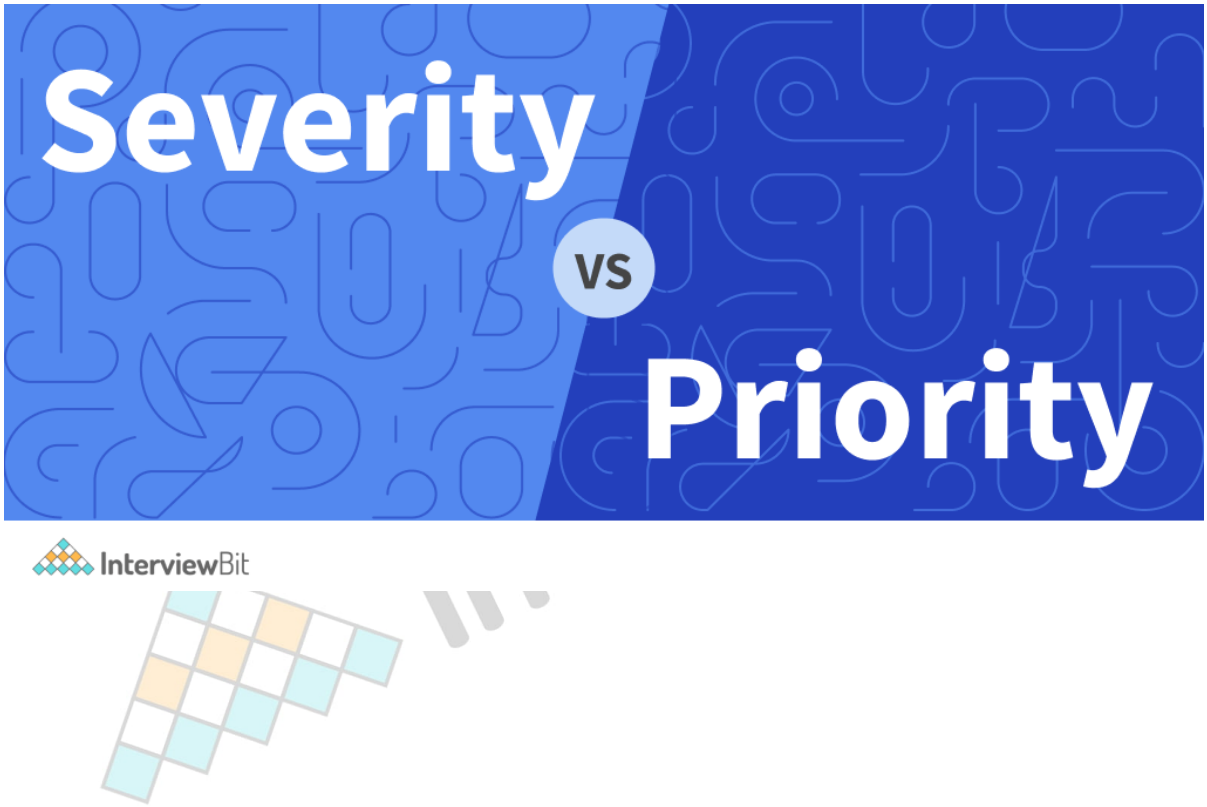
Following are the five dimensions of risks:

- **Schedule:** Unrealistic schedules, the omission of important activities when drafting a schedule, and other factors could stymie project completion on time. If testing is done from a remote place, an unstable communication link can be regarded as a possible danger.
- **Client:** Ambiguous requirements description, lack of clarity on issues, frequent changes to requirements, and other factors could lead to chaos throughout project execution.
- **Human Resources:** Insufficient resources with the required skill level are not available for the project; Resource attrition - Appropriate training schedules for resources must be designed to balance the knowledge level with resources quitting. Underestimating the training effort could have a negative influence on project completion.
- **System Resources:** The inability to obtain or the delay in obtaining all key computer resources, such as hardware and software tools, or software licences, will have a negative impact.
- **Quality:** A combination of issues such as a shortage of resources, a tight delivery timetable, and frequent changes to requirements will have an impact on the product tested quality.

19. What do you understand about severity and priority of a defect in the context of quality assurance? Differentiate between them.

- **Severity:** The severity of a flaw is defined as the degree to which it can affect the software. The severity of a defect is a metric that indicates how serious it is and how much it affects the software's functionality.
- **Priority:** Priority is a parameter that determines which defects should be addressed in what order. The higher-priority defects should be addressed first.

The following table lists the differences between severity and priority:



Severity	Priority
The severity of a software defect is a measure that indicates how serious it is.	Priority is a criterion used to determine which defects should be addressed first.
The degree of severity is proportional to the quality standard.	Priority is linked to the timetable for resolving the issue.
The defect's severity level is determined by the testing engineer.	The defect priorities are set by the product manager.
The severity of a fault refers to how much it affects the functionality.	Priority refers to how quickly a defect must be fixed.
It has objective worth.	It's worth is a matter of opinion.
Its value remains constant over time.	Its value fluctuates over time.
Critical, Major, Moderate, Minor, and Cosmetic are the five levels of severity.	Priority is divided into three categories: low, medium, and high.

20. What do you mean by quality audit in the context of quality assurance?

An audit is an on-site verification activity of a processor quality system, such as inspection or examination. An internal or external quality auditor, or an audit team, conducts a systematic analysis of a quality system as part of a quality audit. Quality audits are conducted at predetermined intervals to ensure that the institution's internal system monitoring methods are well defined and linked to successful action. Audits are an important management tool for confirming objective proof of processes.

In other words, a quality audit is a verification effort aimed at determining the degree of compliance of a product, design, process, or system to a standard specification or procedure. The quality audit includes two parts: the first is an examination of the system in which the products or services are created, known as the quality system audit. The other is a product or service quality audit, which is an examination of the products themselves.

21. How do you figure out how much testing each piece of software requires in the context of Quality Assurance?

The Cyclomatic Complexity can be used to determine how much testing each piece of software requires in our application. The technique aids in the identification of the three questions listed below for the programs/features.

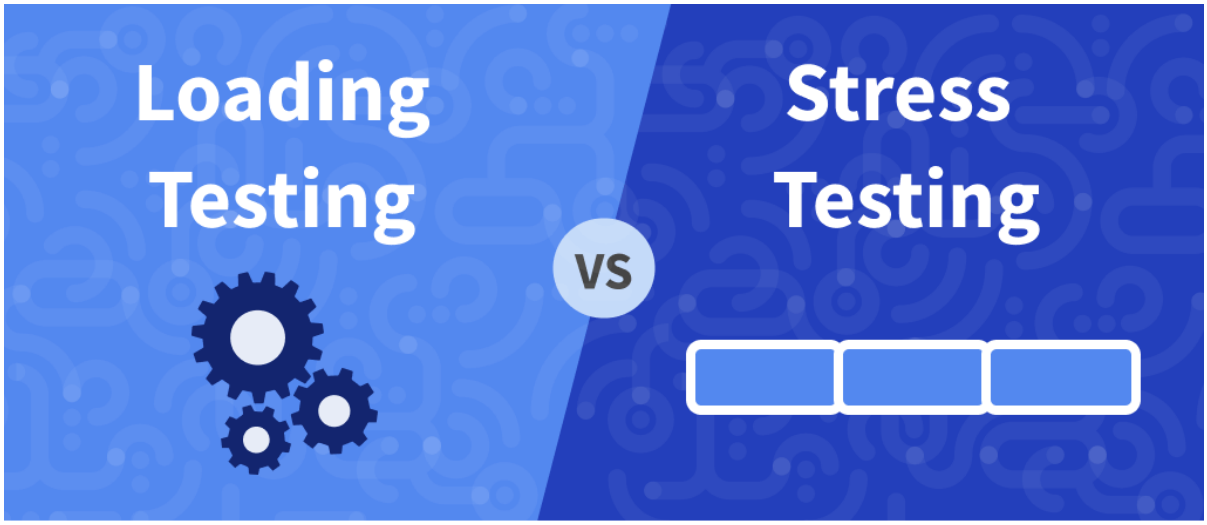
- Is it possible to test the feature or programme?
- Is everyone aware of the feature/program?
- Is the feature/program trustworthy?

We can use this technique to determine the "level" of testing required for our application. It is standard practice to consider a piece of functionality to be complex if the cyclomatic complexity result is greater than or equal to a certain number, and to conclude as a tester that the piece of code/functionality requires in-depth testing. If the Cyclomatic Complexity result is a lesser value, we conclude as QA that the functionality is of lower complexity and adjust the scope accordingly. Understanding the full testing life cycle is critical, and we should be prepared to suggest modifications to our approach if necessary. The goal is to deliver high-quality software, so a QA should take all necessary steps to improve the testing process and the manner the testing team conducts the tests.

22. Differentiate between load testing and stress testing.

- **Load Testing:** Load testing is a type of performance testing that assesses a system's, software product's, or software application's performance under realistic load situations. It also demonstrates how the application would behave when there is increased network traffic on the application and multiple users are accessing the application.
- **Stress Testing:** Stress testing is a sort of software testing that ensures the system's stability and dependability. Under extremely heavy load conditions, this test examines the system's robustness and error handling.

The following table lists the differences between load testing and stress testing:

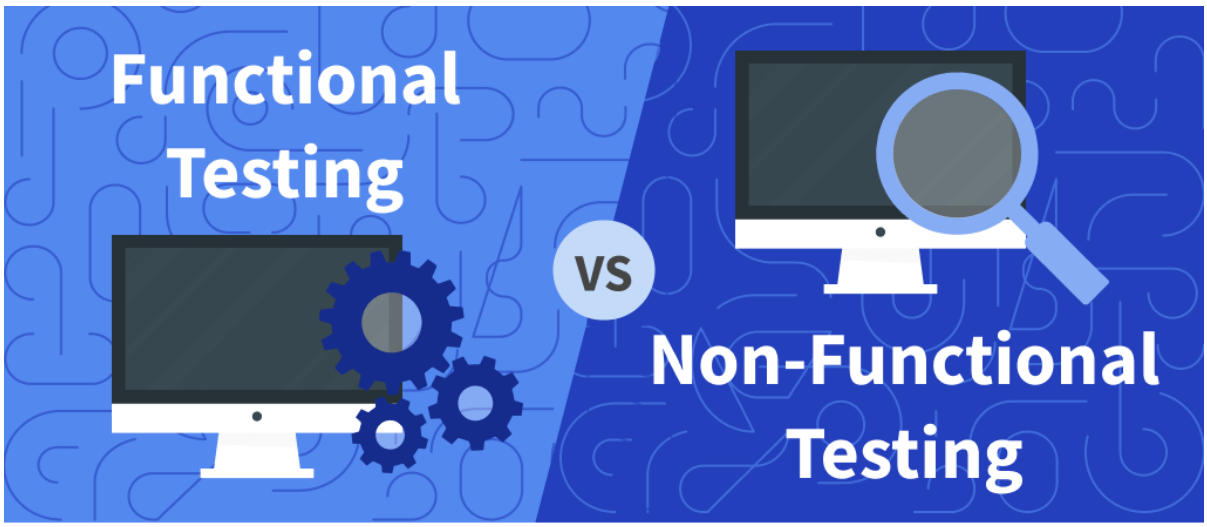


Load Testing	Stress Testing
Load testing is used to set the application's SLA (Service Level Agreement) and see how the system handles a heavy load to determine the system's upper limit.	Stress testing is used to determine the system's or software application's robustness under excessive stress and how it recovers from failure.
The load limit is the point at which a break occurs in load testing.	The load limit in stress testing is higher than the break threshold.
Load testing involves putting the software through its paces with a large number of users.	Stress testing involves putting the system through its paces with varying volumes of data.
Load testing is used to determine a system's or application's maximum capacity.	Stress testing is used to determine how a system behaves when it is put under pressure.
Performance is the factor that is evaluated during load testing.	Robustness and stability are the factors that are examined during stress testing.
Load testing determines a system's or application's operating capacity.	System security is ensured by stress testing.

23. Differentiate between functional and non functional testing.

- **Functional Testing:** Functional testing is a sort of software testing that involves comparing the system to its functional requirements and specifications. Functional testing guarantees that the application meets all of the criteria or standards as specified in the specification requirement sheet. This form of testing is focused on the end product of the processing. It focuses on simulating actual system utilisation but makes no assumptions about system structure.
- **NonFunctional Testing:** Non-functional testing is a sort of software testing that verifies the application's non-functional requirements. It determines whether the system's behaviour is in accordance with the requirements. It examines all aspects that aren't covered by functional testing.

The following table lists the differences between functional and non-functional testing:

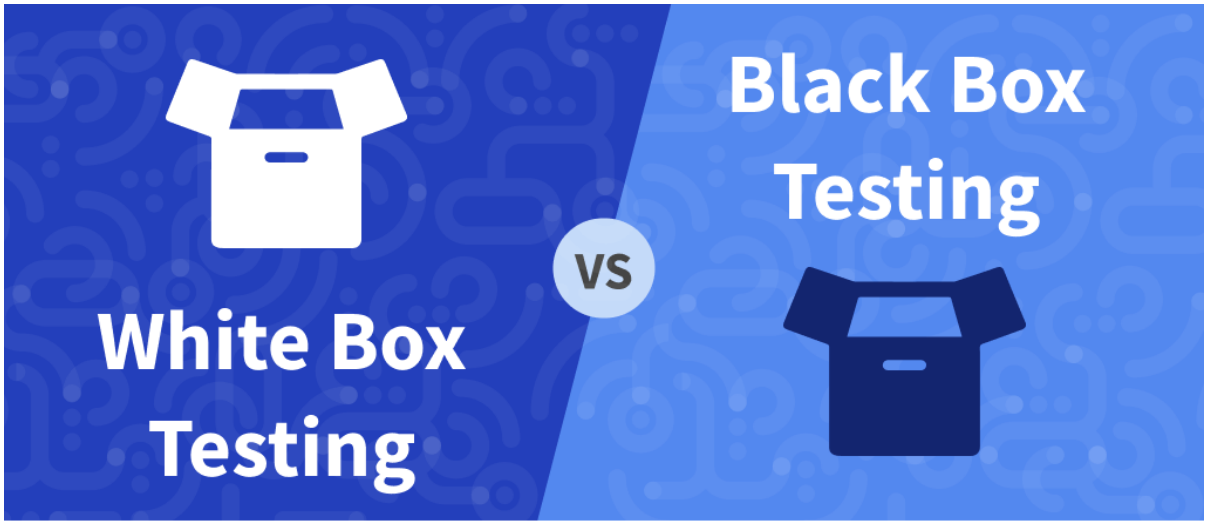


Functional Testing	Non-functional Testing
It verifies an application's activities and actions.	It verifies an application's behaviour.
It is based on the needs of the customer.	It is based on the customer's expectations.
It aids in improving the application's functionality.	It aids in the enhancement of the application's performance.
Manual functional testing is simple to carry out.	Manually performing non-functional testing is difficult.
It evaluates the product's functionality.	It explains what the product is capable of.
The basis for functional testing is the business requirement.	The performance requirement is the basis for non-functional testing.
Example: Unit Testing, Regression Testing, Smoke Testing.	Example: Load Testing, Stress Testing, Performance Testing.

24. What do you understand by black box and white box testing? Differentiate between them.

- **White box testing:** White box testing examines the inside structures of the software, including the employed data structures, internal design, code structure, and how it works, rather than just the functionality, as black-box testing does. It's also known as structural testing, clear box testing, or glass box testing.
- **Black box testing:** Black box testing is a sort of software testing when the software's internal functionality is unknown. The testing is carried out with no knowledge of the products' internal workings. The following methods can be used to conduct black-box testing:
    - **Syntax Driven Testing:** This method of testing is used on systems that may be represented syntactically by a language. Compilers, for example, are a type of language that can be described using a context-free grammar. The test cases are created in such a way that each grammatical rule is applied at least once.
    - **Equivalence partitioning:** Many types of inputs work similarly, so rather than give them all separately, we can group them together and test only one of each group's inputs. The aim is to divide the system's input domain into a number of equivalence classes, each of which works in a similar fashion, i.e., if a test case in one class results in an error, other test cases in other classes will also result in an error.

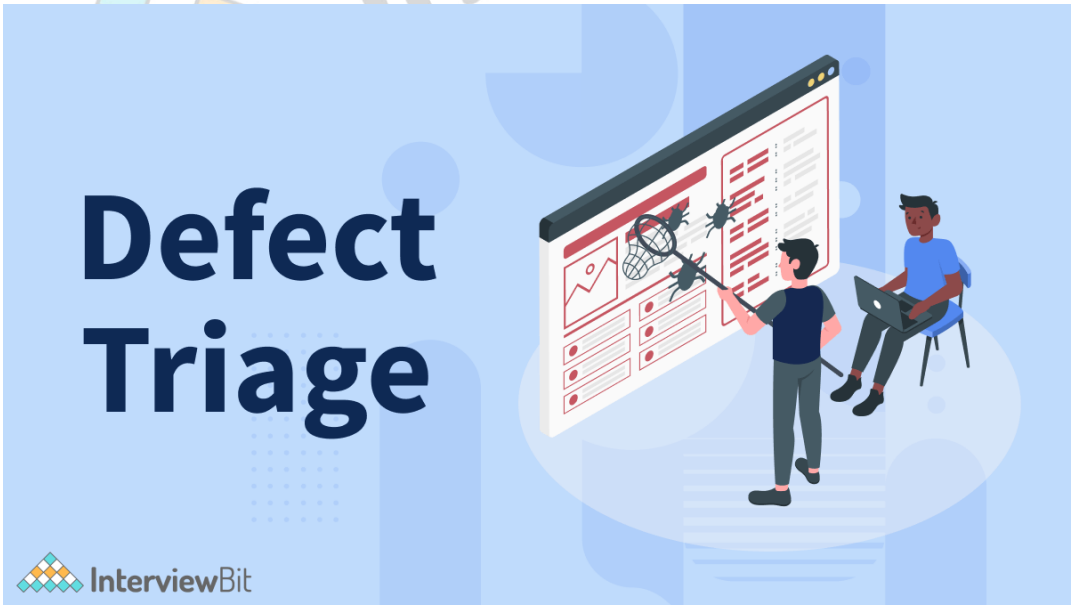
The following table illustrates the differences between white box and black box testing:



White-box testing	Black box testing
It's a testing method in which the tester is aware of the system's internal structure.	It's a method of testing software that doesn't require knowledge of the program's or application's internal structure.
It is also known as Structural testing, clear box testing, code-based testing, or glass box testing.	Data-driven, box testing, data-, and functional testing are all terms used to describe this type of testing.
Because the internal workings of the system are known, the tester may test accordingly.	The application's internal behaviour is unclear, therefore testing is based on exterior expectations.
Testing is better suited to lower-level testing such as Unit and Integration testing.	Higher stages of testing, such as System Testing and Acceptance Testing, benefit from this sort of testing.
White Box testing necessitates a working grasp of programming.	Black Box testing does not necessitate programming knowledge.
It's simple to automate white box testing.	Because the test and programmer are so intertwined, automating them is difficult.

25. What do you understand about bug/ defect triage in the context of quality assurance?

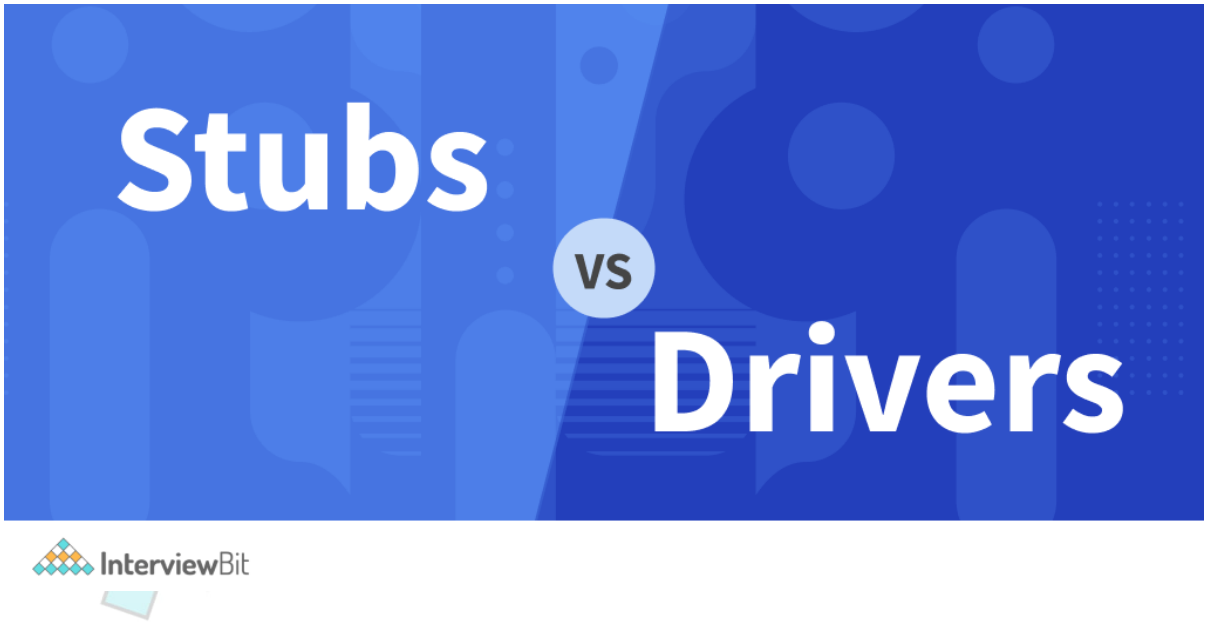
Defect triage is a method of prioritising bugs based on their severity, frequency, risk, and other factors. The term "triage" is used in software testing and quality assurance to describe the severity and importance of new faults. Bug triage's purpose is to review, prioritise, and assign defect solutions. The team must confirm the severity of the fault, make necessary changes, conclude defect resolution, and assign resources. This method is primarily utilised in agile project management. The frequency of the Defect Triage Meeting isn't set in stone. It is dependent on the circumstances of the project.



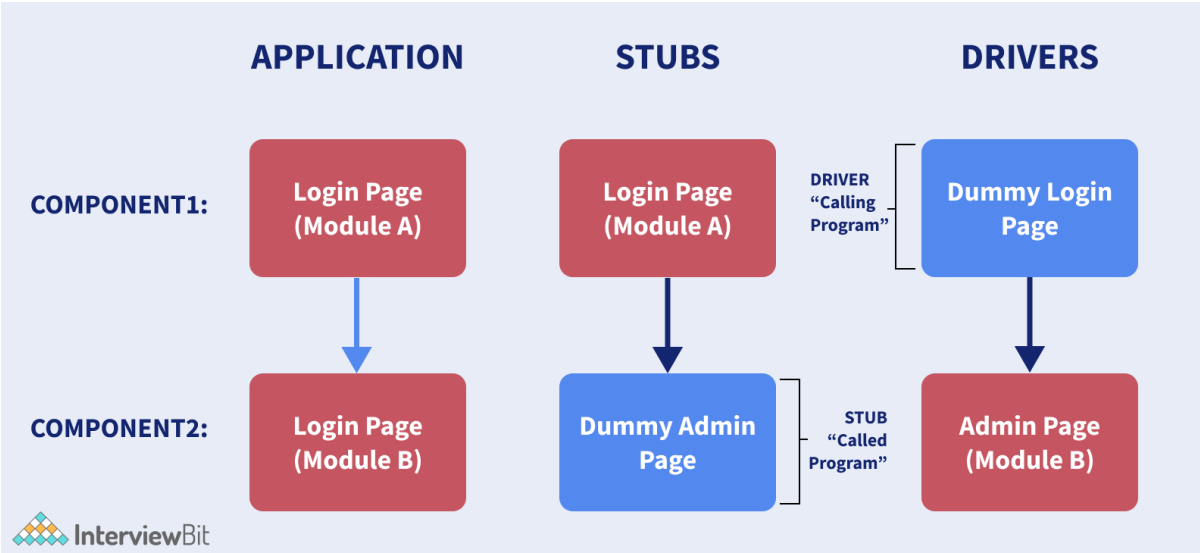
The following are some key elements that influence the frequency of Defect Triage Meetings:

- According to the project's schedule.
- The number of flaws in the system.
- The impact on the availability of team members' schedules.
- The state of the project as a whole.

26. What do you understand about stubs and drivers? Differentiate between them.



- **Stub:** Stubs are created by software developers to be used in place of modules if the respective modules have not been constructed, are missing in the development stage, or are currently unavailable during Top-down testing. A stub is a module that mimics the functionality of an unavailable module. Stubs are used when lower-level modules are required but are currently unavailable. For instance, suppose you have three different modules: Login, Home, and User. Assume that the login module is ready for testing, but the two minor modules Home and User, which are invoked by the login module, are not. At this point, a piece of fake code is written to replicate the Home and User functions. The stubs are the dummy parts of the code.
- **Driver:** Drivers are similar to stubs in that they serve the same goal, but they are more complicated and are utilised in Bottom-up integration testing. Drivers are also used to operate in the absence of essential modules when some modules are absent and unavailable at the time of testing a specific module due to unforeseen circumstances. When high-level modules are missing, drivers are utilised. They can also be used when lower-level modules are missing.
    - Let's use the same example as before. Assume that the User and Home modules are ready to test this time, but the Login module is not. Because the Login module returns data to Home and User, a dummy piece of code is developed to emulate the Login.



The following table lists the differences between Stub and Driver:

Stub	Driver
In Top-Down Integration Testing, stubs are employed.	In Bottom-Up Integration Testing, drivers are used.
Stubs are analogous to software modules that are in the development stage.	Drivers are accustomed to invoking the component that must be tested.
Stubs are primarily utilised when low-level modules are unavailable.	Drivers are mostly used to replace high-level modules, they can also be used to replace low-level modules in specific cases.
They are also referred to as “called programs”.	They are also referred to as “calling programs”.
To test the features and functionality of the modules, stubs are used.	If the core module of the software is not developed for testing, the drivers are used.
If upper-level module testing is completed but lower-level module development is ongoing, the stubs are taken into consideration.	If lower-level module testing is completed while upper-level module development is underway, the drivers are taken into consideration.
Stubs are used to test the main module when lower-level modules are unavailable or in a partially completed state.	When higher-level modules are absent or in a partially built state and we wish to test the lower(sub)-module, we use drivers.

Conclusion

A rigorous quality assurance strategy yields a software product that is both high-quality and error-free. However, quality assurance entails a lot more. Quality assurance is vital for many parts of the business, including client interactions and the company's reputation in the market, in addition to finding flaws and places for improvement in a product.

Useful Resources:

Software Testing

Automation Testing

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions