Functional Testing Interview Questions

To view the live version of the page, [click here.]()

Contents

Functional Testing Interview Questions for Freshers

1. Why is functional testing important?
2. What are different types of functional testing?
3. Explain how functional testing is performed or what are the steps to carry out functional testing.
4. State difference between functional and non-functional testing.
5. Explain unit testing vs functional testing.
6. What do you mean by functional testing vs regression testing?
7. Explain Adhoc testing.
8. How does 'Build' differ from 'Release'?
9. What is the main difference between Monkey testing and Adhoc testing?
10. State difference between Alpha and Beta testing.
11. What are the different Test Techniques used in Functional testing?
12. Explain risk-based testing and what important factors are needed to be considered in risk-based testing.

Functional Testing Interview Questions for Experienced

13. Explain equivalence partitioning.
14. What do you mean by boundary value analysis?
15. State difference between functional and structural testing.
16. What do you mean by UFT (Unified Functional Testing)?
17. What do you mean by Data-driven testing?
18. Explain Smoke testing and Sanity testing.

Functional Testing Interview Questions for Experienced

(.....Continued)

19. What is RTM (Requirement Traceability Matrix)
20. Why is RTM (Requirement Traceability Matrix) important?
21. Difference between Retesting and Regression testing.
22. What do you mean by Defect Severity and Defect Priority?
23. What do you understand by the term accessibility testing?
24. What is build acceptance testing?
25. Why automate functional tests? What should we look for while choosing the right automation tool?
26. Explain what are functional test cases.
27. How should Test Cases be written? What are the important points to consider?
28. List out some examples of functional test cases.
29. What are some possible login features that need to be tested in any Web application?
30. What is the best way to ensure functional test cases cover all product areas?
31. What test cases are to be automated?

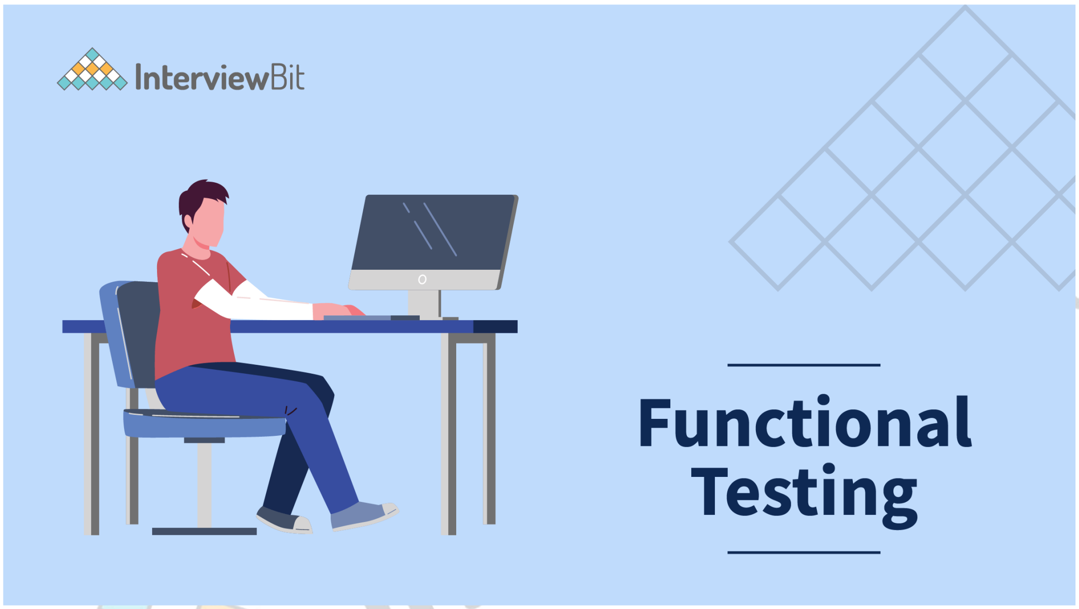
Let's get Started

Functional testing is crucial to the software development process and is a fundamental requirement to assess how the software system functions. At first glance, this sounds easy and simple - and it can be. Nonetheless, functional testing is usually more detailed and involved when the situation is complex.

Let's take a look at some of the deeper concepts of functional testing in this article. But first, let's define what functional testing actually is.

What is Functional Testing?

A software system/application consists of many components that must all work together, and also individually, in order to work properly. These components must be thoroughly tested for functionality and this can only be done through functional testing. As part of the QA (**Quality Assurance**) process, functional testing is a fundamental requirement to assess how the software system functions. Each software component is first checked for its expected output, then tested twice to ensure that its output does not impact the rest of the system. Functional testing does not consider the source code of the application during testing, as it is a type of black-box testing.



Functional testing is often performed at various stages of software development. The purpose of functional testing is not only to validate a software system or component against the various functional specifications and requirements defined but also to verify whether it is ready to be deployed into the live environment. These questions can help you identify areas in which your skills can be valuable and can help you prepare for challenges ahead.

Functional Testing Interview Questions for Freshers

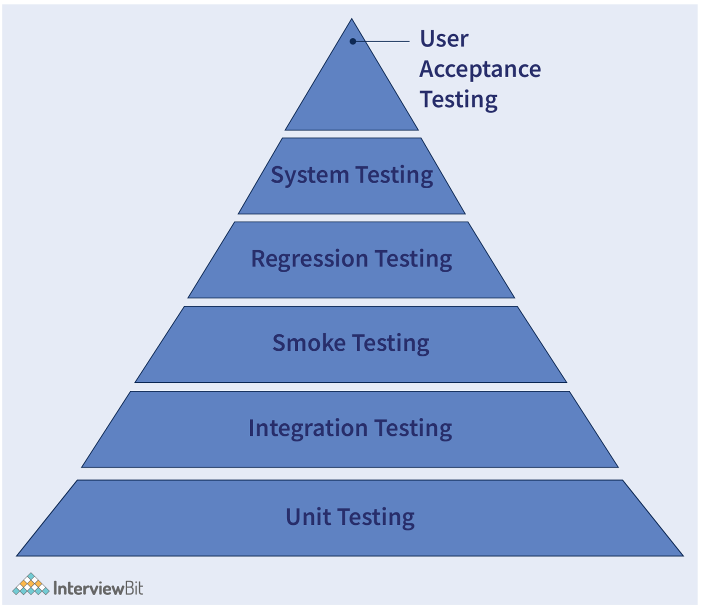
1. Why is functional testing important?

Validating/Testing the quality and functionality of software requires functional testing. The purpose is as follows:

- Ensures the delivery of high-quality products.
- A quality assurance team uses this **software testing** technique to ensure software functionality is in line with the Software Requirements Specification (SRS) and to ensure the system functions as intended by the user.
- In essence, this testing method is utilized for testing user-friendliness features, specific error conditions, accessibility features and other important features of the software.
- This allows testers to verify that the software they test works as intended and gives developers feedback about how to improve software products.
- Functional testing is crucial to determine whether your code is working correctly or if errors are likely to arise in the future.

2. What are different types of functional testing?

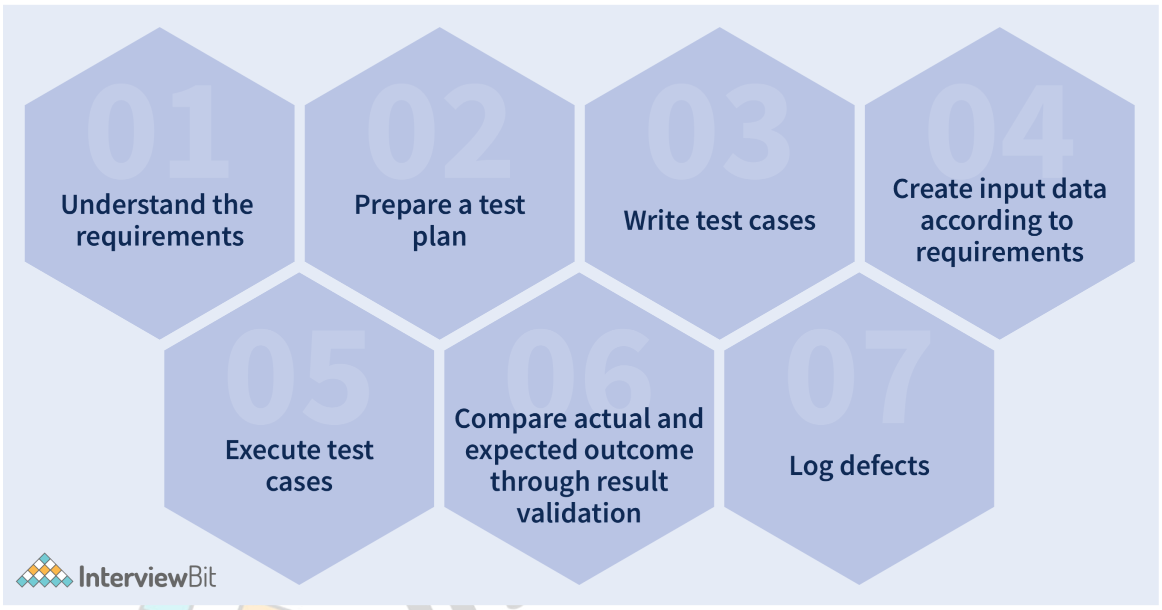
Ultimately, functional testing is aimed at ensuring that software works as specified and meets user expectations. Functional testing may seem simple on its face, but it involves a variety of methods, some of which may be preferred or prioritized over others based on the application and organization. These methods are outlined below:



- **Unit testing:** It involves testing the individual units/blocks of code of the software to validate that they perform as intended.
- **Smoke testing:** This test only examines the basic functionality of a system to ensure the software works properly (or is not plagued with too many problems) so that the next test can proceed.
- **Sanity testing:** This type of testing is also known as a build-verification test and is usually performed after a smoke test. The testing is performed after a complete software build with minor changes is released to verify that the code changes introduced continue to work as intended.
- **Integration testing:** This test combines individual components/units and tests them together. Tests will ensure that they interact appropriately, as well as reveal any faults between the integrated units if present.
- **Regression testing:** This type of testing makes sure that alterations to the code won't affect the system's functionality. The purpose of this test is to ensure that adding fresh code, improvements, or fixing bugs does not cause instability or compromise the software functionality.
- **User Acceptance testing:** It involves testing software systems from the user's perspective to ensure they meet the requirements. Before software is released to market or production, it must pass the user acceptance test.

3. Explain how functional testing is performed or what are the steps to carry out functional testing.

When performing functional testing, you should follow the steps below as outlined in the following diagram:


01
Understand the requirements
02
Prepare a test plan
03
Write test cases
04
Create input data according to requirements
05
Execute test cases
06
Compare actual and expected outcome through result validation
07
Log defects
InterviewBit

- **Understand the requirements:** Prior to conducting functional testing, it is necessary to thoroughly understand the business requirements. Understand the requirements document specification and clarify any doubts and queries if there are any.
- **Prepare a test plan:** An effective test plan is critical for successful functional testing. Test engineers need to focus on the objectives of tests so that planning and organizing test activities will become easier.
- **Write test cases:** Write the test cases considering the software requirements and keeping all of the test scenarios in mind.
- **Create input data according to requirements:** Testers need to prepare input data to test specific functionality of software applications. When this type of data is used to test an application, it is referred to as test input data. From these inputs, determine what the expected outcome should be.
- **Execute test cases:** Next, the test input data is used to execute the test cases.
- Compare actual and expected outcomes through the result validation: In this step, the actual test results are compared with the expected results to determine whether the test has passed or failed.
- **Log defects:** The variation of the difference between the actual and expected output is referred to as a log defect. It is the task of the testers to log defects so that the development teams can fix them as soon as possible.

4. State difference between functional and non-functional testing.

The two major **types of software testing** are functional testing and non-functional testing.

- **Functional Testing:** This type of testing verifies that each feature/function of a software application works as intended. A functional test can only be passed if a software system has all its functions working as intended.
- **Non-Functional Testing:** This type of testing examines non-functional aspects like performance, usability, compatibility, reliability, etc., of a software system. This process verifies whether the system behaves as expected or not.

Functional vs Non-Functional Testing:

TESTING
Functional Testing
VS
Non-Functional Testing
InterviewBit

Functional Testing	Non-Functional Testing
It verifies that the actions and operations of an application perform as expected.	It verifies how an application behaves.
Business requirements are taken into account when functional testing is done.	Non-functional testing is performed according to customer expectations and performance requirements.
This testing determines whether the results are consistent with expectations.	This testing measures the speed and response time of a software application under specific conditions.
Customer feedback assists in reducing the risks associated with a product.	Feedback from the customer is more valuable for non-functional testing because it allows for improvement and enables the tester to learn the customer's expectations.
Testing tools such as manual and automation are used to perform functional testing first.	Manual testing can be problematic due to the non-functional testing process, which takes into account scalability, reliability, speed, and other performance parameters.
Functional Testing Examples: Unit Testing, Smoke Testing, Integration Testing, Regression Testing, Sanity	Examples: Performance Testing, Load Testing, Stress Testing, Scalability Testing, Volume Testing, etc.

5. Explain unit testing vs functional testing.

Testing software or applications aims to build a quality product. Functional testing and unit testing are the backbones of software testing.

- **Unit Testing:** In its simplest form, unit testing entails testing individual components or units within the software. In this step, each unit of code is validated and checked to determine if it is performing as you expected it to.
- **Functional Testing:** This type of software testing evaluates a system's functionality against the functional requirements. Each software component is first checked for its expected output, then tested twice to ensure that its output does not impact the rest of the system.

Unit Testing Vs Functional Testing:

Unit Testing	Functional Testing
In this type of test, the smallest units or modules are tested individually.	It verifies that an application performs as expected.
It is possible to find issues that crop up frequently in modules by running unit tests.	A functional test identifies the issues preventing an application from performing correctly. This may include scenario-based issues as well.
There is no chance for the issue to escape.	As the number of tests to run is always infinite, there is a higher chance of issues escaping.
In other words, it is a form of white-box testing.	Essentially, it is black-box testing.
A unit test is fast and can help write clean code, but it cannot assure that the app will work as intended. It shows us where errors are in the code, though.	A functional test takes time and effort but ensures the system will meet functional requirements. The test identifies any problems or defects in the functionality.

6. What do you mean by functional testing vs regression testing?

- **Functional testing:** It involves checking that an application's functions work as expected. It is usually a black box testing process. The purpose of functional testing is to make sure the product designed by developers meets the requirements as specified by stakeholders.
- **Regression testing:** This process is conducted once a software build has been released. The purpose of this test is to ensure adding fresh code, improvements, or fixing bugs does not cause instability or compromise the software functionality.

7. Explain Adhoc testing.

The term Adhoc Testing, also known as random testing, generally refers to a type of testing that occurs without proper planning or documentation. Adhoc testing has

- No documentation
- No Test cases
- No Test Design

Ad-Hoc Testing
No documentation
No Test Design
No Test Case
InterviewBit

Ad hoc testing is usually performed randomly without documentation or testing design and it is usually unplanned. Ad hoc Testing does not adhere to any particular structure and is done randomly on any part of the application to identify defects/bugs. When time is limited and exhaustive testing cannot be performed, adhoc testing may be conducted. The tester needs to have a thorough understanding of the system under test in order to conduct effective adhoc testing.

Example: Adhoc testing is cost-effective and can save you a lot of time; one example would be when the client needs the product by 4 PM today, but the development will be finished by 2 PM. With only 2 hours to work with, the developer and tester team can test the system as a whole by taking some random inputs and checking for bugs.

8. How does 'Build' differ from 'Release'?

- **Build:** Once the source code for the software application is developed, developers convert it into standalone or executable form (build). As soon as the build has been completed, the development team passes it on to the testing team for testing. The software testing team checks this build for multiple bugs and if it fails to meet the requirements, the build is rejected. A build occurs prior to the release, and it is generated more frequently. During software development, several builds are generated.
- **Release:** At the end of development and testing, the release is the final product. As soon as the software is tested and certified, it is delivered to the customer. It is possible, however, for a release to have multiple builds. As such, it refers to the software developed after the testing and development phases are completed.

9. What is the main difference between Monkey testing and Adhoc testing?

There are two different types of software testing that can be run on the software: monkey testing and adhoc testing. Tests are conducted to ensure that the system is bug-free.

- **Adhoc Testing:** It is usually performed randomly without documentation or testing design and it is usually unplanned. The tester needs to have a thorough understanding of the system under test in order to conduct effective adhoc testing. It is meant to ensure that the application or system does not crash.
- **Monkey Testing:** This is similar to Ad Hoc Testing, except that it can be done by testers without any prior knowledge or information about the software. The monkey test is an automated test conducted without planning any specific test in advance. A tester tests the system by randomly trying its features to see if he can break it.

10. State difference between Alpha and Beta testing.

- **Alpha Testing:** This is a type of internal acceptance testing performed by the company's software quality assurance team/developers. Testing is done to identify bugs before releasing a product to end-users or to the public.
- **Beta Testing:** It is the last phase following the internal full alpha test cycle, which is a form of external user acceptance testing. In this phase, companies release the software to a few outside groups other than their own test teams and employees. As a result, beta testing can be summarized as testing conducted by real users in a real environment.

Alpha vs Beta Testing: Learn More

Alpha Testing	Beta Testing
Testers within the organization are responsible for performing Alpha Testing.	Beta testing is carried out by a few individuals/clients/users outside the organization.
During Alpha Testing, reliability and security are not thoroughly tested.	While Beta Testing, the product's availability, security, and robustness are examined.
Blackbox and Whitebox testing are both included in Alpha Testing.	Blackbox testing is mostly involved in Beta testing.
Issues and bugs encountered in Alpha Testing are immediately addressed and fixed.	The majority of the issues and feedback from the beta testing will be implemented in the next software product version.
During Alpha testing, quality is ensured before moving on to Beta testing.	During beta testing, the focus is not only on the quality of the product but also on gathering users' feedback and assessing whether it is ready for real-world usage.
Before launching the product into the market, it undergoes alpha testing.	During the marketing of a software product, beta testing is conducted.

11. What are the different Test Techniques used in Functional testing?

Functional testing involves two distinct test techniques, which can be defined as follows:

- **Requirement based testing:** This type of testing prioritizes the requirements based on risk factors. Furthermore, this will ensure that all critical test paths are covered in the testing process.
- **Business process-based testing:** This type of functional testing examines the business process. Testing scenarios are based on knowledge of the business process. Examples include service delivery and deployment, etc.

12. Explain risk-based testing and what important factors are needed to be considered in risk-based testing.

Risk-based testing refers to the process of prioritizing tests according to risk, which is used as a basis for developing a Test Strategy. An organization can use risk-based testing (RBT) to prioritize testing software features and functions according to the probability of failure, the importance of the feature, and the impact of a failure. Testing is then performed, starting with the highest risk. Testers who use a risk-based approach are more likely to be aware of risk factors that can lead to project failures.

Risk-Based Testing
An Effective Technique to Manage Risks Efficiently in Software Testing
TESTING
InterviewBit

Following are the main factors to be considered in risk-based testing:

- Identification of when and how risk-based testing should be implemented on an appropriate application.
- To determine which measures are effective in detecting as well as handling risks in critical areas of the software application.
- Achieving the project outcome while balancing risk and quality.

Functional Testing Interview Questions for Experienced

13. Explain equivalence partitioning.

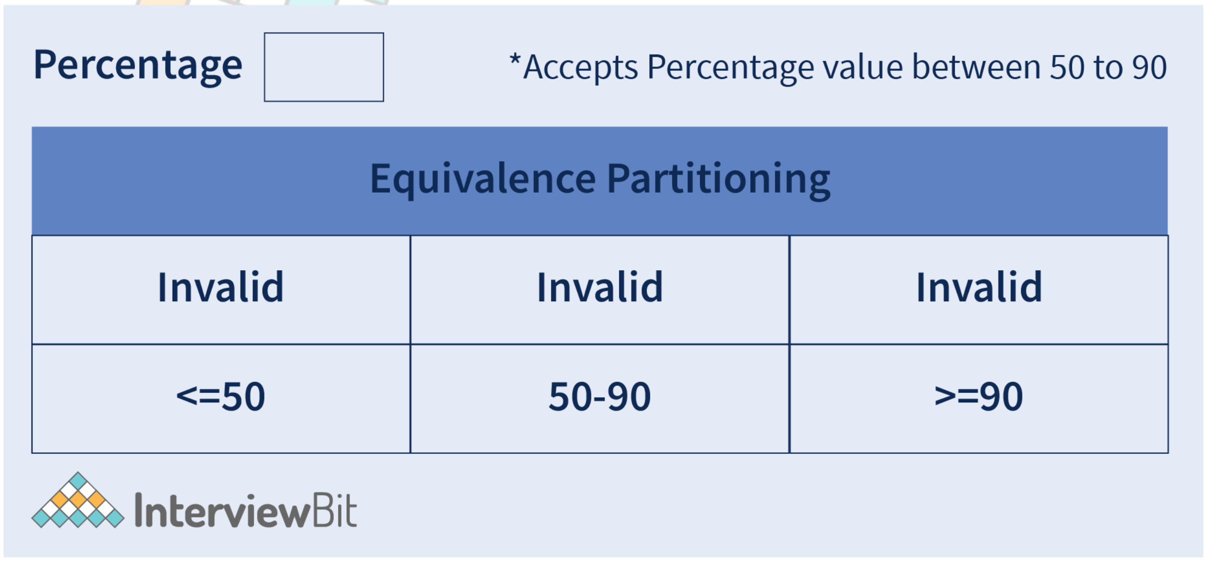
Equivalence Partitioning is also called Equivalence Class Partitioning (ECP) and is a form of black-box testing. In this method, input domain data is divided into equivalence classes (partitions) and test cases are derived using these classes of data. Then, while testing, one sample value is picked from each class. By using this method, test cases are generally reduced to a finite set of testable cases that still cover the maximum requirements.

A technique of equivalence partitioning is applied only when input data values can be divided into ranges. For each range partition, only one condition will be tested, assuming that all other conditions within the same partition will behave similarly.

Example: Let's say you have an input field that can accept only percentage values between 50 and 90%. In that case, it would be pointless to write thousands of test cases for 50-90 valid input numbers, plus others for invalid data.

The Equivalence Partitioning method outlined above can be used to divide test cases into three classes of input data. Each test case represents a class. As you can see, in the example provided above, we were able to split our test cases into three equivalence classes (can be more) composed of invalid and valid inputs.

Test cases for input box accepting percentages between 50 and 90 using equivalence partitioning:



Percentage

*Accepts Percentage value between 50 to 90

Equivalence Partitioning		
Invalid	Invalid	Invalid
<=50	50-90	>=90

- **The valid partition:** Between 50%-90% (The expectation is that the text field would handle all input percentages between 50%-90%, the same way).
- **The first invalid partition:** <=50% (We'd expect the text field to reject all values greater than or equal to 50%).
- **The second invalid partition:** >= 90% (We'd expect the text field to reject all values greater than or equal to 90%).

14. What do you mean by boundary value analysis?

The boundary value analysis is a technique for testing the boundary value of an equivalence class partition. A boundary value analysis identifies errors at the boundaries, opposed to within ranges in equivalence partitioning.

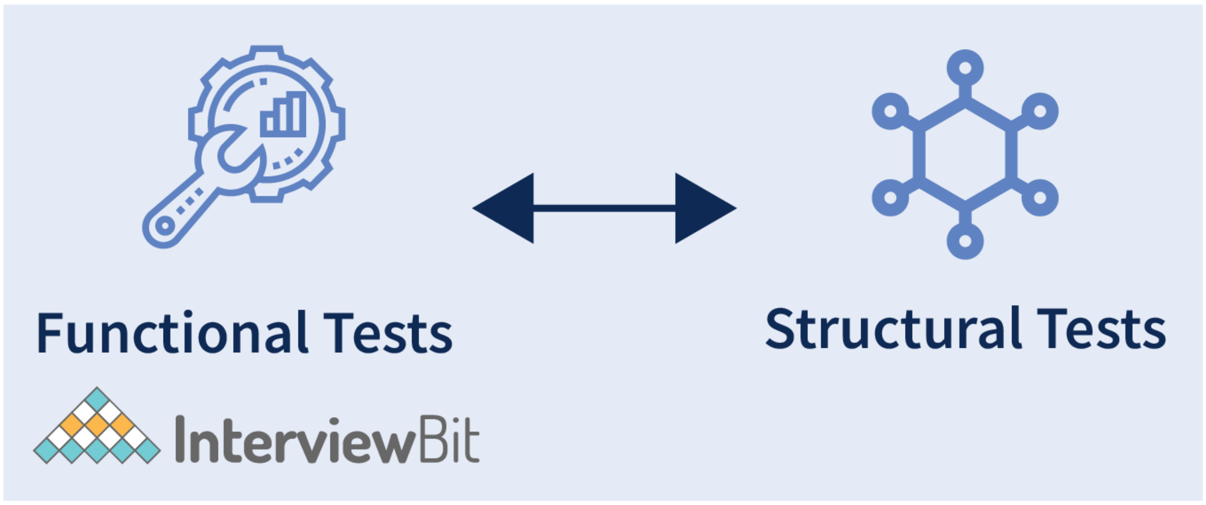
Example: Consider an input field in an application that can accept a minimum of 5 characters and a maximum of 10 characters. We were able to split our test cases into three equivalence classes composed of invalid and valid input. Then 5-10 is considered as valid and <4 and >10 is considered as invalid.

Test cases for application input field accepting numbers between 5-10 using boundary value analysis:

- **The valid partition:** Between values 5-10 (For this test case, we will use the same test data as the input boundaries of the input domain, namely values 5 and 10).
- **The first invalid partition:** <5 (The test data value for this case will be just below the edge/boundary of the input domain, i.e., value 4).
- **The second invalid partition:** >10 (The test data value for this case will be just above the edge/boundary of the input domain, i.e., value 11).

15. State difference between functional and structural testing.

Functional testing differs from structural testing in the following ways:



Functional Testing	Structural Testing
The purpose of functional testing is not only to validate a software system or component against the various functional specifications and requirements defined but also to verify whether it is ready to be deployed into the live environment.	This test evaluates a software's internal design or the structure of its code. It is also called white-box testing, clear box testing, or glass box testing.
Functional testing is a black-box test since no knowledge of the internal logic of the system is used to create test cases.	Functional testing is a black-box test since no knowledge of the internal logic of the system is used to create test cases.
Some types of functional testing include system testing, regression testing, sanity testing, and user acceptance testing.	Some types of functional testing include system testing, regression testing, sanity testing, and user acceptance testing.

16. What do you mean by UFT (Unified Functional Testing)?

UFT (Unified Functional Testing), also known as QTP (QuickTest Professional), is an automated functional testing tool that helps testers to conduct automated tests to identify errors, defects, and other deviations from the expected behaviour of a software application.

- Specifically, it is used for functional, regression, and service testing.
- This tool runs UI-based test cases and also automated non-UI test cases, such as database testing and file operations.
- It is compatible with Windows platforms and multiple browsers like Firefox, Chrome, etc.
- It supports a wide variety of software development environments, including SAP, Oracle, etc.
- Also, it allows for Quality Assurance checks to be run on the software being tested.
- This tool provides easy navigation, validation of results, and report generation. etc.

17. What do you mean by Data-driven testing?

A data-driven testing approach is a method of functional testing where a series of test scripts are executed repeatedly with the use of data sources like Excel, CSV files, Spreadsheets, XML files, and SQL databases. Data sources like these are used as inputs for generating output. Next, the output is compared to what was expected to verify the system or software.

A data-driven approach is preferable because testers often have multiple data sets for a single test, and it can be time-consuming to create individual tests for each data set. By using data-driven testing, data and test scripts can be separated, and the same test script can be run for different combinations of input data, resulting in efficient testing results.

Example: Let's assume we want to test the login system with 100 different data sets and multiple input fields. We can have the below three approaches:

- **Approach 1:** Write 100 scripts, one for each dataset, and run each test individually.
- **Approach 2:** Change the value in the script manually and run it several times.
- **Approach 3:** Import Excel data and pull test data one by one from Excel rows, and then run the script.

The first two scenarios listed are arduous and time-consuming. As a result, it would be best to utilize the third approach (data-driven testing).

18. Explain Smoke testing and Sanity testing.

- **Smoke testing:** This test only examines the basic functionality of a system to ensure the software works properly (or is not plagued with too many problems) so that the next test can proceed.
- **Sanity testing:** This type of testing is also known as a build-verification test and is usually performed after a smoke test. The testing is performed after a complete software build with minor changes is released to verify that the code changes introduced continue to work as intended.

Smoke vs Sanity Testing: Learn More

Smoke Testing	Sanity Testing
Smoke testing is done to assure that the acute functionalities of the program are working fine.	After receiving a software build that has undergone minor code or functionality changes, sanity testing is conducted to ensure that bugs have been fixed and that there are no further problems that can be introduced by the changes.
It is possible to perform smoke testing manually or with the help of automation tools.	It is common to perform sanity testing manually, not through automation.
Both developers and testers perform smoke testing.	Testers perform sanity testing.
There is documentation or scripting for smoke testing.	There is no documentation for sanity testing.

19. What is RTM (Requirement Traceability Matrix)

The Requirement Traceability Matrix, or RTM, is a tool that keeps track of requirements as a system or application progress through a testing process. As soon as the requirements document is received, the RTM is created and maintained until the system or application is released. RTM is used to ensure that all requirements in the requirements specification have been implemented before the release of the system.

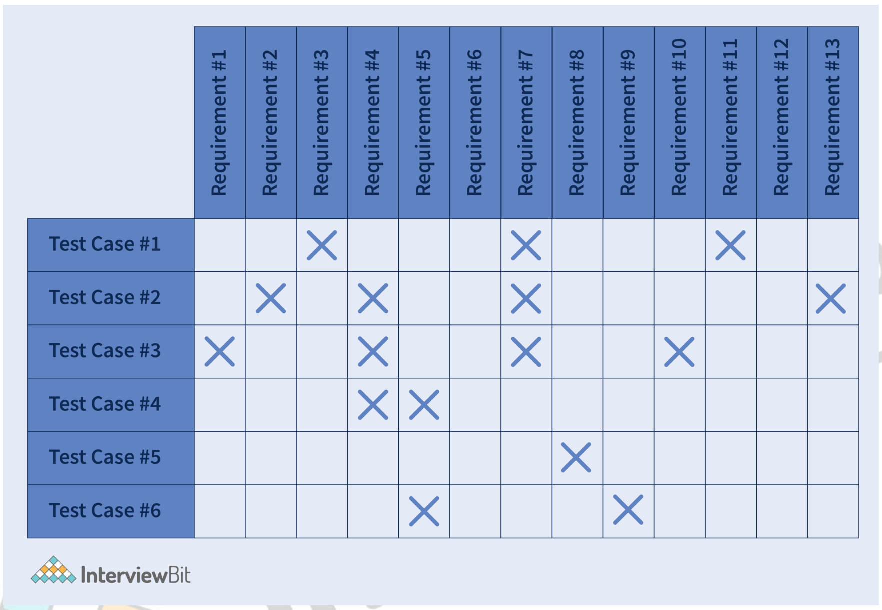
20. Why is RTM (Requirement Traceability Matrix) important?

Each tester should be responsible for understanding the client's requirements and ensuring that the output product is error-free. To accomplish this goal, the QA team must create test cases after thoroughly analyzing the requirements. As a result, the client's software requirements need to be divided further into different scenarios and finally into test cases. Each of these cases needs to be tested separately.

Here is a question: how can one make sure that the requirement is tested in all possible scenarios? Are any requirements left out of the testing process?

The simplest way is to trace the requirements to their corresponding test cases and scenarios and it is termed as 'Requirement Traceability Matrix.' Typically, the traceability matrix is a worksheet that contains the requirements and their associated test cases and scenarios as well as the current status of those tests, whether they were successfully executed or not. This will aid the testing team in understanding the extent to which testing has been done for the specific product.

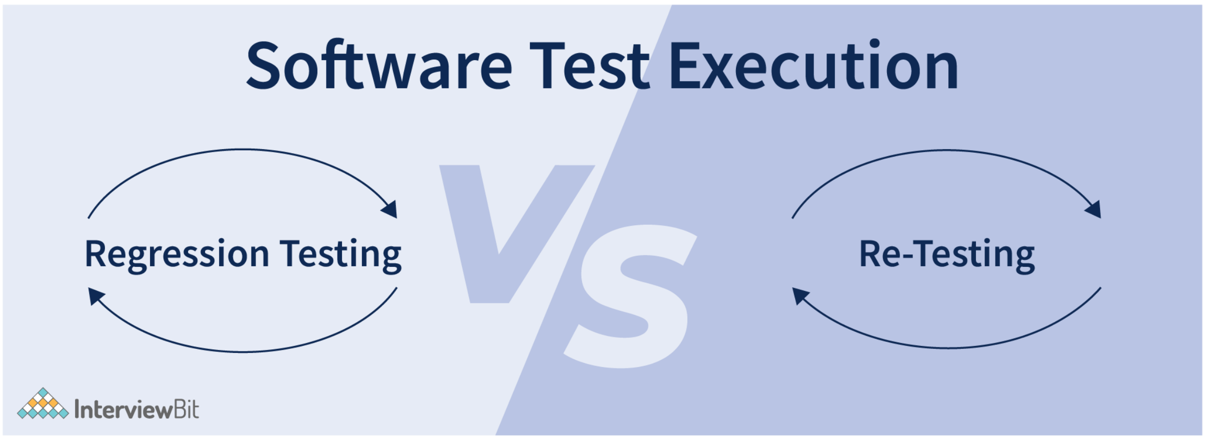
For example, The following is an example of different test cases that have different requirements to be tested. A matrix shows the requirements as columns on top and the test data as rows. The "X" here indicates which test cases relate to which requirements.



	Requirement #1	Requirement #2	Requirement #3	Requirement #4	Requirement #5	Requirement #6	Requirement #7	Requirement #8	Requirement #9	Requirement #10	Requirement #11	Requirement #12	Requirement #13
Test Case #1			X				X				X		
Test Case #2		X		X			X						X
Test Case #3	X			X			X			X			
Test Case #4				X	X								
Test Case #5								X					
Test Case #6					X				X				

21. Difference between Retesting and Regression testing.

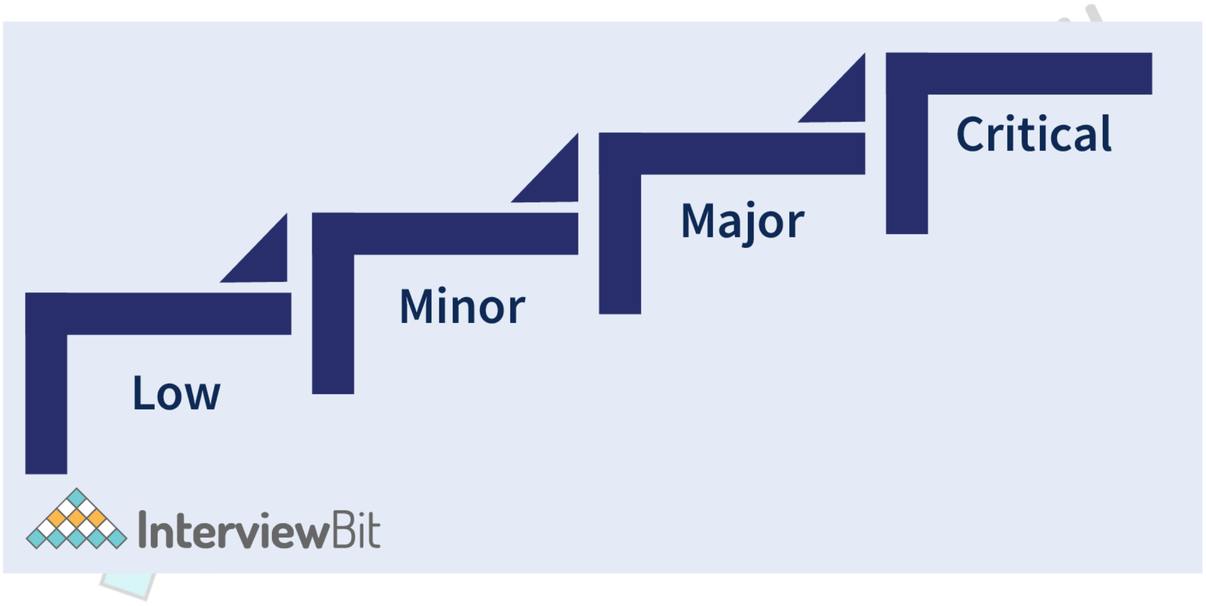
Regression testing differs from re-testing in the following ways:



Retesting	Regression Testing
During retesting, testers make sure that all test cases that failed during the last execution are passed after the defects are fixed.	This type of testing makes sure that alterations to the code won't affect the system's functionality.
Upon fixing the defects, re-testing is carried out.	The purpose of this test is to ensure that adding fresh code, improvements, or fixing bugs does not cause instability or compromise the software functionality.
Retesting involves defect verification.	Regression testing does not include defect verification.
Retesting cannot be automated.	Regression testing can be automated because it is time-consuming and expensive to do manually.
Test cases that failed generically are subjected to this type of testing.	Test cases that pass generically are subjected to this type of testing.

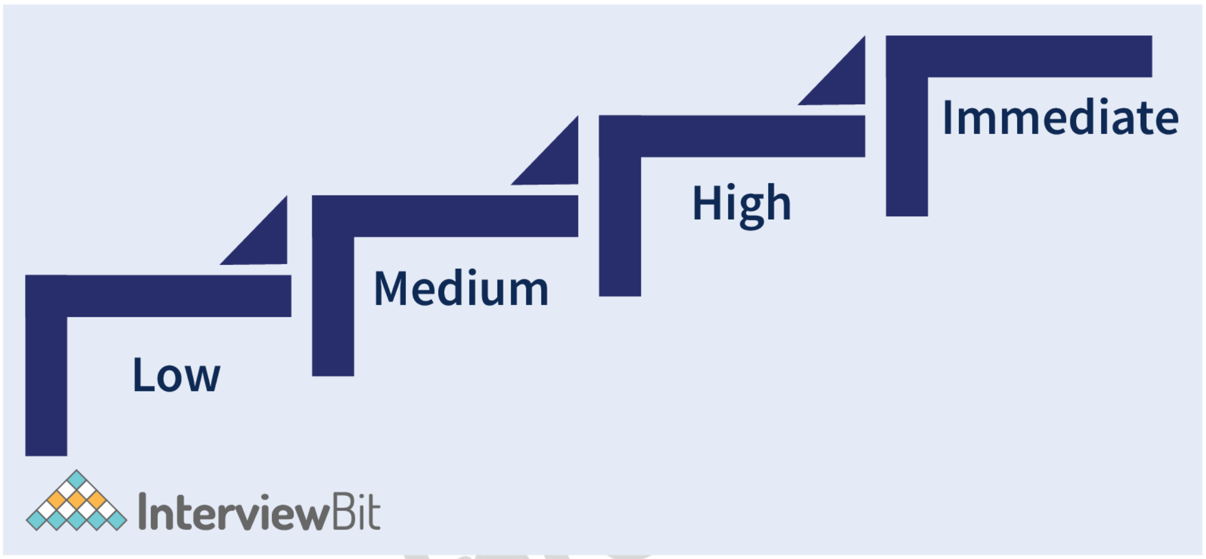
22. What do you mean by Defect Severity and Defect Priority?

Defect Severity: This is the degree or extent to which a defect impacts the application being tested. The severity of the defect is a measure of its impact on the software's functionality. A defect with a higher severity level will have a greater impact on the application. Defect severity is classified into four categories:



- Critical
- Major
- Medium
- Low

Defect priority: It is a parameter determining the order in which defects must be fixed. A high priority defect will more likely result in an unusable or stuck application, and it should be corrected as soon as possible. Defect priority is classified into three categories:



- High
- Medium
- Low

23. What do you understand by the term accessibility testing?

Accessibility Testing refers to the process of testing the usability of a software application by ensuring that it can be accessed by people with all abilities and disabilities. The intent of this testing is to verify both the usability and the accessibility of the application. The accessibility of the application should allow for people with all types of disabilities including:

- Visual Impairments
- Physical Impairment
- Hearing Impairment
- Cognitive Impairment
- Learning Impairment

In the present scenario, the web has acquired a major position in our lives with e-commerce sites, e-learning, e-payments, etc. Therefore, everyone should be able to use technology to a greater degree, especially people with disabilities. Accessibility testing can be performed both manually and automatically.

24. What is build acceptance testing?

BAT (Build Acceptance Testing), also known as BVT (Build Verification Testing), is a type of software testing intended to ensure the most important functions are working properly when new code is implemented. Based on the results of this testing, a software build can be considered stable enough to continue for further testing. Basically, it's a set of tests that are run on each new build in order to verify that it conforms to the requirements of the build before sending it to the testing team for further examination. BAT processes are typically automated. In the event BAT fails, then that build will be assigned to a developer for the fix once again.

Advantages:

- Enhanced coverage and efficiency of testing.
- Testers are less concerned with the possibility of unstable builds.
- All builds are tested daily, so any major issues can be caught early.
- Testing is automated, so there is no need for manual intervention.

25. Why automate functional tests? What should we look for while choosing the right automation tool?

Automating functional tests can certainly save time and effort. Testing can be done continuously without human intervention. Furthermore, human errors can be reduced, which prevents bugs from slipping through the test phase. It is especially useful during the development phase because it helps you find bugs and problems earlier, increasing your team's efficiency. As there are various automation tools available, it is essential to identify the right automation tool for the job.

Now the question is how do you choose the right automation tool? The following points should be taken into account when choosing an automation tool.

- All members of your QA team should be able to use the tool easily.
- It must be able to operate in different environments seamlessly.
- The tool should support the reusability of test cases in case the UI is changed.
- Ensure that it has features specific to your team's needs. If, for instance, your testing team needs specific reporting or logging, or if it needs to use different scripting languages, then the tool must have these features as well.

26. Explain what are functional test cases.

Every feature/function of a piece of software should be tested thoroughly before it's released, and many of them should be continually tested. The functional test cases are the documents written by QA managers for the purpose of conducting functional testing. Functional test cases inspect the software's functionality across a range of actions or conditions to ensure the desired outcome. Following are the features of a functional test case:

- Name or description of the function/feature.
- The preconditions for testing.
- The steps needed to test.
- The expected outcome/result.

All of these provide the tester with everything they need to satisfy the test case requirements. It is also a good idea to write the location of the function in the description section, especially if the app is very large and complex, or if the same function is found in some different areas of the app.

27. How should Test Cases be written? What are the important points to consider?

Having understood what functional test cases are, let's take a look at how they're written. Below is a list of points you should consider while writing test cases:

- The first step is to have a broad overview of the testing project and determine what should be tested.
- Now it's time to write the test cases. Before writing test cases, it is important to understand the client's requirements clearly. According to the requirement document specifications, every functional and nonfunctional requirement should be covered, from UI interfaces to compatibility. Normally, a traceability matrix is maintained to ensure that all requirements are implemented and tested.
- It is a good idea to periodically check test cases to ensure that they are not redundant.
- While writing a test case, priority is an important consideration. In this way, the tester is able to begin testing the application with the high priority test cases, followed by the medium and then the low priority test cases.
- Test cases should be written in a simple language and in a structure that is easy to understand. For test cases, the input data values should be valid and within a wide range.

28. List out some examples of functional test cases.

Here is an example of functional test cases:

- **The new task to be assigned:** Consider the case of assigning a new task to an appropriate user.
- **Description:** Create a new task and assign it to an appropriate user.
- **Preconditions:** When the user gets assigned, he/she should be enabled to receive email notifications.
- **Steps:**
    - Create two separate user profiles, each with an email address.
    - Create a new task using one of the accounts.
    - Assign the task to your other user account.
- **Expected result**: The user should receive an email confirming that they are assigned a task.

29. What are some possible login features that need to be tested in any Web application?

Listed below are the possible scenarios you can use to test an application's login feature:

- Input fields, such as username and password, should be validated both with valid and invalid values.
- You can also try entering a valid email address with an incorrect/invalid password, as well as entering an invalid email address with a valid password. Verify that a proper error message appears.
- Get logged in to the application by entering valid credentials. Check if you are still logged in by closing and reopening the browser.
- Once the user has logged in, navigate again to the login page to see whether he or she needs to log in again.
- If you are signed in through one browser, open the application through another browser to verify that you are signed in through the other browser.
- You should change your password after logging into the application and then try to login again with the old password.

30. What is the best way to ensure functional test cases cover all product areas?

The most likely scenario is that you need to conduct regression testing in order to cover parts of the product that may be impacted by any new updates or product releases. The tester, in addition to testing the product's core features or functionality, should also list any areas that are affected by this function or prone to breaking due to high usage and complexity of coding.

In order to get a fresh perspective on test coverage, test cycles should be scheduled over two or three days. When strategizing and writing test cases, testers should open the app so they can ensure they aren't overlooking a function or feature they would have missed otherwise.

31. What test cases are to be automated?

Automation enables the avoidance of repetitive manual work, faster feedback, and a reduction of time spent running tests repeatedly. Despite this, automated testing cannot cover all test cases, so determining which test cases to automate is important.

It can be difficult, however, to figure out which tests need to be automated since it is pretty much dependent on the functionality of the software product. Here are some of the parameters to select test cases for automated testing.

- Different data set for the test case.
- Execution of a test case with a different browser.
- Execution of test cases in different environments.
- Execution of a test case with complex business logic.
- Execution of the test case with different users.
- Test case with a lot of data, etc.

Conclusion:

Functional testing is a quality assurance procedure used to evaluate whether a software application or component conforms with its stated functional requirements. It is clear that functional testing is essential to building a top-quality software product.

The purpose of functional testing interviews is to assess candidates' skill sets and determine whether they are suited for the position. Throughout this article, we've covered 30+ functional testing interview questions with answers, which will help you nail the interview. Hopefully, you found the article informative and helpful.

Useful Resources:

- Automation Testing Interview Questions
- API Testing Interview Questions
- Database Testing Interview Questions
- Performance Testing Interview Questions

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