Top 50 Manual Testing Interview Questions and Answers in 2023

By Great Learning Team Updated on Nov 16, 2022 14030

Software testing is a process that takes place during the software development life cycle to ensure that the software is accurate and meets the requirements. In any software development project, testing is the most important step.

In today's competitive environment, testing is crucial to the growth of any software product. Manual tests are important in software development because they can be used when automated testing is impossible. As a result, there is still a high demand for persons with manual testing abilities. This article is about Manual Testing Interview Questions that will help you learn software testing.

The fundamental purpose of software testing is to ensure that the software being tested generates the correct result for a given input.

When learning about software testing, it's crucial to remember that testing does not enhance programme quality on its own. Or that a large amount of testing does not imply high-quality software. Testing is a quality indicator, providing critical input to the software's creators, allowing them to take the required steps to correct any issues discovered during testing.

The interviewer for a Software Tester or Quality Assurance (QA) role usually asks the following interview questions. Let's find out some of the most asked questions' answers on Manual Testing:

1. How does quality control differ from quality assurance?

Quality Control: Quality control is a product-oriented approach, a "part of quality management focused on fulfilling quality requirements". It also runs a program to detect if there are any defects and as well it assures that the software meets all of the requirements put forth by the stakeholders

Quality Assurance: Quality assurance is defined as a process-oriented approach that is a "part of quality management and focused on providing confidence that quality requirement will be fulfilled". It mainly focuses on assuring that the methods, techniques, and processes used to create quality deliverables are applied correctly.

2. What is Software Testing?

Software Testing is a process used to verifying the correctness, identifying errors, completeness, missing requirements versus the actual requirements and quality of developed software. For that, it contains a series of activities that are conducted with an intention of

finding errors in software and make them corrected before the product is released to the market. It basically checks all the requirements are working fine and available in it.

3. Why is Software Testing Required?

Software testing is a very important and mandatory process. Software testing makes sure that the software product is safe and have all that which it should have, basically it checks does the product fulfils the needs and requirements of the users and make sure that the product is good enough to be released to the market. Following are some reason which proves that software testing is important:

- It checks whether if there any defects and errors that were made during the development phases.
- It reduces the time for coding cycles by identifying issues at the initial stage of the development.
- Software testing makes sure that software application requires low maintenance and results in a more accurate, consistent and reliable manner.
- Testing ensures that the customer finds the organization reliable and satisfactory in the application performance and maintenance.
- Makes sure that software is bug-free and fulfils all the requirements and makes sure quality matches the market standards.
- Ensures that the application doesn't stop working or face any failures in working.

4. State the difference between manual testing and automation testing?

Manual Testing	**Automation Testing**
The accuracy & reliability of test cases is low in manual testing, as they are performed by humans so they are more likely to have errors in it.	Automated testing, on the other hand, is more re as automated tools are used to perform tests.
Manual testing is time-consuming as human resources perform all the tasks.	The execution in automation testing is faster tha manual as software tool execute the tests
Investment in manual testing is low, but Return of Investment(ROI) is low as well.	Both the Investment cost and Return of Investm are high in automation testing.
Manual testing is generally preferred when the test cases run once or twice. It is also suitable for Exploratory, Usability and Adhoc Testing.	Automation testing is generally used for Regres Testing, Performance Testing, Load Testing or h repeatable functional test cases

Manual testing requires human observation to find out any issues. Therefore manual testing is better in improving the customer experience.	In automation testing, there is no human observa required, that's why there is no guarantee of a po customer experience.

5. Write the two main categories of software testing?

Software testing is a huge domain. It can be broadly categorized into two main categories of software testing such as:

- **Manual Testing** – This is a software testing process. The test cases are tested manually without any automated tool. It is the oldest type of software testing that can find both hidden and visible defects of the software. In this testing, the software application is tested manually by QA testers.
- **Automation Testing** – In this software testing process the test cases are tested using a special automated testing software tool. In this automation testing assistance of tools, scripts and software are used to perform test cases by repeating pre-defined actions. Automation testing focuses on replacing manual testing which includes human activities with the systems or devices that enhance efficiency.

6. What is quality control?

Quality control is a product-oriented approach of running a program to ensure the quality of the product is maintained or improved if it has any defects, they fix them as well as making sure that the software meets all of the requirements needed by the users.

7. What makes a human a good test engineer?

A software test engineer is any professional who ensures that the product meets all the expectations and fulfil all the requirements. A software test engineer basically creates a process that would test a particular product in the software industry.

- A good test engineer should understand the priorities and should have the ability to take the point of view of the customer
- Should be passionate for quality and attention to minute details
- A good test engineer should have the ability to assert his ideas or opinions to maintain a cooperative relationship with developers
- Ability to communicate in a manner via which he can report negative things in a positive way with both technical (developers) and non-technical (customers, management) people
- Prior experience and flexibility to support whenever it is required in the software development industry is always a plus point
- Ability to take a risk whenever it is required at the time of application by judging the situations and make important decisions

8. What different types of manual testing are there?

There are six types of manual testing:

- Black Box Testing: In this level of testing, testing is done without interfering with any of its internal structure and workings.

- White Box Testing: In this level of testing the internal structure and working is tested.
- Unit Testing: In this level of testing the individual units or parts are tested.
- System Testing: In this level of testing the software product is completely tested
- Integration Testing: In this level of testing the individual units and components are combined together and are tested
- Acceptance Testing: In this level of testing the software product is tested whether it has met all requirements or not.

Taking up manual testing courses will help you gain an in-depth understanding of the concept.

9. What is the role of documentation in Manual Testing?

Documentation plays a crucial role in achieving effective software testing. All the Details like requirement specifications, designs, configurations, code changes, test plans, business rules, inspection reports, test cases, user manuals, bug reports, etc. should be documented.

Documenting the test cases will make it easier for developers to assess the testing effort developer will need along with test coverage, tracking and tracing requirement. Some commonly applied to investigate procedures of the documentation associated with software testing are:

1. Test Plan
2. Test Scenario
3. Test Case
4. Traceability Matrix

10. What are the phases involved in Software Testing Life Cycle?

The different phases of the software testing life cycle are:

Phases	**Explanation**
Requirement Analysis	QA team first learn the requirement in terms of what they will be testing & calculate th testable requirements.
Test Planning	In this phase, the test strategy is made. The objective & the scope of the project is defin
Test Case Development	In this phase, detailed test cases are defined and evolved. For testing the data is prepare the team.

Test Environment Setup	It is arranging software and hardware as per the requirements of the testing teams to ex test cases.
Test Execution	It is the process of executing the code and checking the expected results with the actual results.
Test Cycle Closure	It gives a summary of all the tests conducted during the software development, include testing team member meeting & assessing cycle completion criteria based on test cover quality, cost, time, critical business objectives, and software.

11. Explain the difference between alpha testing and beta testing.

- **Alpha Testing** – Alpha Testing is a type of user acceptance testing in which software testing is performed to identify bugs and errors before releasing the product in the market or to the users.
- **Beta Testing** – Beta testing is quite opposite of the alpha testing. In this testing, the product is tested by the real users in the real environment and according to the needs the changes are made in it. It is also a type of acceptance testing.

12. What are the different levels of manual testing?

There are Four levels of manual testing are:

- **Unit testing** – Unit testing is referred to the testing of the smallest piece or the individual units of code that can be logically isolated in a system. It is mainly focused on the functional accuracy of the software product.
- **Integration Testing** – In Integration testing, the individual units or components are combined together as a group. This testing is used to test whether the combined units or components are working as they intend to when integrated The main aim here is to expose the faults in the interface between the modules.
- **System Testing** – In system testing all the components of the software are validated as a whole in order to make sure that the overall product meets the requirements specified. It evaluates all the end to end system specifications for accurate results. There are many types of system testing, including usability testing, regression testing, and functional testing.
- **User Acceptance Testing** – It is the last level of testing, acceptance testing, or UAT (user acceptance testing), which is performed after the software is completely tested. It is used to determines whether or not the software is ready to be released.

13. What is a testbed in manual testing?

An environment configuration for testing is known as a testbed. It is creating an environment suitable for testing an application, it includes the hardware as well as any software needed to run the program and to be tested. The testbed consists of specific hardware, operating system, software, network configuration, the product under test, other system software.

14. Explain the procedure for manual testing?

The manual testing process is consists of the following steps:

- Planning a test and Controlling a test
- Analysis of the requirements.
- Test cases Implementation and Execution
- Evaluating exit criteria and Reporting the bugs
- Checking completion of the test

15. When should you opt for manual testing over automation testing?

There are many cases when manual testing is best suited over automation testing, like:

- **Short-time projects:** Automation testing requires high investment and planning, this testing is aimed at time saving and saving resources. But still takes time and resources to design and maintain them and the expenses are also high. For example, if you are creating a small promotional website, it can be much more efficient and easy to depend on manual testing.
- **Ad-hoc Testing:** In ad-hoc testing, there is not any specific approach. Ad-hoc testing is totally performed without planning and documentation where the understanding and insight of the tester is the only important factor. Which can be achieved by using manual testing.
- **Exploratory Test:** In this type of testing approach it mostly depends upon the tester's knowledge, experience, analytical, logical skills, creativity, and intuition. So human involvement is a must in exploratory testing. So manual testing is very suitable here.
- **Usability Testing:** While performing usability testing, it is measured by the tester how user-friendly, efficient, or convenient the software or product is for the end-users. In usability testing Human observation plays the most important part, so manual testing sounds seems very appropriate.

16. What is the test case?

A test case is a document that has all the requirements on it like a set of conditions or actions, the inputs, procedures that are performed on the software application in order to check whether it fulfils all the given requirements or not.

Test cases are a concept that includes a specific idea that is to be tested, without specifying the exact steps to be taken or data to be used. For example, in a test case, you document something like '***Test if a discount can be applied on actual price***'.

17. What is API testing?

API testing is a type of software testing which involves testing application programming interfaces (APIs) to determine if they meet expectations for functionality, reliability, performance, and security. In simple words, API testing is planned to find out bugs, inconsistencies or deviations from the expected behaviour of an API. Commonly, applications have three separate layers:

- Presentation Layer or user interface
- Business Layer or application user interface for business logical processing
- Database Layer for modelling and manipulating data

API testing is performed at the most critical and important layer of software architecture, the Business Layer.

18. What is the difference between verification and validation in testing?

Verification	Validation
It is a technique, in which documents, design, code and program is checked in order to ensure the requirements are fulfilled or not. Here testing is done without executing the code. The verification process mainly includes activities like reviews and inspection.	It is a dynamic mechanism of testing and validat where testing is done by executing the code and check the software product actually meets the customer needs or not. Validation includes activ like functional and non-functional testing techni

19. What is the difference between a bug and a defect?

When the software or applications are not working as per the requirements then these faults are called a bug, it occurs during testing time. A defect is an irregularity between expected results and actual results, it is detected by the developer after the product release in the market.

20. What are the advantages of manual testing?

Advantages of manual testing are:

- This testing is cheaper in cost when compared to automated testing.
- Manual testing is easy to learn and it needs less time.
- The product analysis from the customer end is possible only with manual testing
- GUI testing can be done more precisely with the help of manual testing as visual accessibility and precedences are difficult to automate
- It is highly suitable for small applications, short-term projects where the test scripts are not going to be repeated or reused more times
- It is very suitable when the project is especially at the early stages of its development
- Highly reliable, since it is done manually where automated tests can contain errors and missed bugs.

21. Explain the defect life cycle

A **defect life cycle**, also known as Bug Life Cycle is a journey in which a defect goes through various phases during its whole lifetime. The cycle starts when a defect is found and ends when a defect is solved, after ensuring that it will not occur again

22. What is regression testing? When to apply it?

“Regression Testing is basically retesting a previously tested program to ensure that it still performs the same after making changes in it and no defects have been occurred or uncovered in unchanged areas of the software, this process is known as regression testing.”

When some small changes are made in some part of the system, a regression test is used to ensure that the system's functionality does not break somewhere in the system. The regression test makes sure that the changes do not change the performance and working of the system. When the following events occur it is recommended to perform regression testing:

- When new functionalities are added to the system
- In case there are some changes in code or any other part of the system
- When the is a defect fix in the system.
- When the performance is not up to the mark.
- In case of environmental changes
- When there is a particular segment fix

23. What is the difference between a bug, a defect and an error?

Bug – A bug is a fault in the software which is occurred during the testing time. They come up due to some coding error and leads a program to malfunction. They can also cause a functional issue in the product. These are fatal errors that could break the functionality and can result in a system crash, or cause performance failure.

Defect – A defect is a difference between the expected results and actual results, it is detected by the developer when the product is live. The defect is an error found when the application goes into production. Basically, it refers to several problems or issues of the software products, with its external behaviour, or with its internal programs.

Error – An error basically refers to a mistake, misunderstanding, or misconception, on the part of a software developer. The developers are divided into various categories includes software engineers, programmers, analysts, and testers. For example, a design notation is misunderstood by the developer, or a variable name is incorrectly written by the programmer – leads to an error. An error normally occurs in software, and due to which there is a change in the functionality of the system.

24. What are the drawbacks of manual testing?

Drawbacks of manual testing are:

- The main drawback is it quite risky and suspectable because of the human factor, humans make mistakes. So it is not reliable.
- Some of the test types like load testing and performance testing are not possible to test manually
- Regression tests can be really time-consuming if they are done manually
- Manual testing has very limited scope when compared to automation testing
- Manual testing is not useful in very large organizations and in time-bounded projects. Due to more time consumption.

25. What is the difference between system testing and integration testing?

System Testing	**Integration Testing**

System Testing tests the software application as a whole to ensure if the system is complementing the user requirements	Integration testing tests the interface between modules of the software application
Involves both functional and non-functional testing like sanity, usability, performance, stress and load	Only functional testing is performed to check whether the two modules when combined give right outcome
It is high-level testing accomplish after the integration testing	It is low-level testing performed after unit testi

26. What is the test harness?

A test harness is the collection of software and test information that is put together to test a program unit by running it under varying environments like stress, load, data-driven, and monitoring its behaviour, response and outputs. Test Harness is divided into two main parts:

1. A Test Execution Engine
2. Test script repository

27. What is test closure?

Test Closure is a document that gives a summary of all the tests that are conducted during the software development life cycle and it also gives a detailed analysis of the removed bugs and errors found. This document contains a report of test cases executed, total no. of experiments executed, total no. of imperfections discovered, add total no. of imperfections settled, total no. of bugs not settled, total no of bugs rejected and so on.

28. What is the pesticide paradox? How to overcome it?

According to the *pesticide paradox*, if the same tests are carried out over and over again, then an outcome of this, the same test cases will no longer be able to find new bugs. Developers will be extra careful in those parts where testers found more defects and might not look into other areas.

Methods to prevent pesticide paradox are following:

- To continually write a whole new and different set of test cases to exercise different parts of the software.
- To regularly review the existing test cases and add a new test case to them.

Using these methods, it's possible that we can find more defects in the segment where defect numbers dropped.

29. Define what is a critical bug

A critical bug is a bug that has got the tendency to impacts a major functionality of the given application. This means a large area of functionality or major system component is completely broken and there is no method to overcome this problem to proceed forward. The Application cannot be given to the end-user unless the critical bug is fixed.

30. What is the difference between Positive and Negative Testing?

Positive Testing	**Negative Testing**
Positive testing ensures that your application working as expected. If any error occurred during positive testing, then the test fails	**Negative testing** determines that your applica can handle the invalid input or unwanted user behavior
In this testing, only a valid set of values are checked by the tester.	In this testing, Testers apply all their creativity validations to the application against unwante

31.What is Defect Cascading in Software Testing?

A defect that is induced by another defect is known as defect cascade. In this case, one problem in the application triggers the occurrence of another defect. Hide to further phases without being recognised when a problem is present in any step but not identified. As a result, the number of defects will rise.

Let’s have a look at a visual representation of this.

You’re working on a web page’s login module.

Phase 1 You’re developing the Register User Module for Login in phase one, and while the Mobile number is required, you can leave it blank due to a glitch and it will go undiscovered.

Phase 2 entails creating a login form with a username and password, with the password being an OTP given to the user’s registered mobile number.

Now that the Register module has a defect in which the mobile number can be left blank, this could result in a login failure or even a system error or crash if the null mobile number is not handled properly.

This is a case of defect cascading.

32. What is the term ‘quality’ mean when testing?

To find out what is meant by ‘quality’ in Quality Assurance (QA), first let’s understand what is Quality Assurance (QA). Quality assurance testing is a quality assurance (QA) or quality testing technique that ensures a company’s products or services are of the highest quality.

Organizations must also ensure that their processes for delivering the intended results meet certain quality benchmarks, as QA attempts to offer consistent results through a set of defined procedures.

In a nutshell, quality assurance (QA) encompasses all efforts centred on developing standards and procedures for ensuring that software meets a set of requirements before it is published to the public.

The important thing to remember is that QA does not entail product testing. Instead, it concentrates on the techniques in order to achieve the best results. In the end, QA activities are process-oriented. It’s all about establishing quality standards and putting in place the necessary checks and balances to ensure that the final product fulfils those criteria.

Organizations must first develop a measurable set of quality measures, as well as a procedure for verifying that those metrics are reflected in software. This means that businesses must obtain a thorough understanding of what the end-user considers to be a “excellent experience.”

These metrics must be properly defined so that the software testing team can obtain data and identify what needs to be changed. The software’s internal quality (code) must be honed to perfection before the exterior quality (the end-user experience) can be maximised.

It’s all about establishing quality standards and putting in place the necessary checks and balances to ensure that the final product fulfils those criteria.

Quality software is generally free of bugs, delivered on time and on budget, fulfils specifications and/or expectations, and is maintainable. But, then again, the term “quality” is a subjective one. It depends on who the ‘client’ is and how influential they are in the greater scheme of things. Each type of ‘client,’ for example, will have their own definition of ‘quality’ — the accounting department may describe quality in terms of revenues, while an end-user may define quality as user-friendly and bug-free.

33. What is black box testing, and what are the various techniques?

Black-Box Testing, also known as specification-based testing, examines a software/functionality application’s without knowing anything about the item’s internal structure or design. The goal of this testing is to ensure that the system as a whole function properly and meets user expectations. The following are some examples of black-box testing techniques:

- Analysis of Boundary Values
- Partitioning by equivalence

- Transitional State Testing
- Decision Table Based Technique
- Graph-Based Testing
- Error Detection Method

1. **Analysis of Boundary Values**

It is a type of black-box testing that is frequently utilised and also serves as the foundation for equivalency testing. Boundary value analysis puts software to the test by using extreme test data values in test cases. BVA is used to discover defects or errors caused by data restrictions in the input.

Boundary value analysis puts software to the test by using extreme test data values in test cases. BVA is used to discover defects or errors caused by data restrictions in the input.

2. **Partitioning by equivalence**

This test case design technique splits the input into comparable classes and checks the input and output. It must be tested at least once to ensure that the data is adequately tested. It is the most thorough type of testing, and it also decreases input redundancy.

In the example above, taking inputs for a test case data will have three classes, one of which will be tested.

3. **Transitional State Testing**

This testing approach makes advantage of the system's inputs, outputs, and state during the testing process. It compares the programme to the test data's sequence of transitions or occurrences.

It tests for behavioural changes of a system in a certain state or another state while keeping the same inputs, depending on the type of software being tested.

A login screen, for example, will allow you to enter your username and password three times before rejecting you. The user will be directed to the login page for each invalid password. The user will be taken to an error page after the third try. This state transition approach takes into account the system's numerous states and inputs in order to pass just the proper testing sequence.

4. **Decision Table Based Technique**

This technique produces test cases from various contexts. It evaluates multiple test scenarios in a decision table structure, where each condition is evaluated and satisfied, in order to pass the test and produce accurate results. When there are various input combinations and possibilities, it is preferable.

When placing an order, for example, food delivery software checks various payment methods as input and makes a decision based on the table.
Case 1: If the end-user has a card, the system will skip checking for cash or coupons and go straight to placing the order.

Case 2: If the customer has a coupon, no credit card or cash will be checked, and action will be taken.

Case 3: if the end-user has cash, action will be taken.

Case 4: No action will be performed if the end-user is unable to supply anything.

5. **Graph-Based Testing:**

Graph-based testing is a type of testing that uses graph

It's similar to a decision-based test case design technique in that it considers the relationship between input instances and linkages.

6. **Error Detection Method:**

This approach of developing test cases includes taking accurate predictions about the output and input in order to remedy any system flaws. It depends on the talents and judgement of the tester.

This method compares and validates the findings using two different versions of the same software.

34. What is white box testing, and what are the various techniques?

White-Box Testing, also known as structure-based testing, necessitates a deep understanding of the code. The goal of this testing is to improve security, input/output flow across the programme, and design and usability. The following are some examples of white-box testing methods:

Statement Coverage: This approach dictates that every single statement in the code be evaluated at least once throughout the software engineering testing process.

Branch Coverage: This method explores every possible path that a software programme may take (if-else and other conditional loops).

In addition to the above-mentioned coverage types, there are also Condition Coverage, Multiple Condition Coverage, Path Coverage, and Function Coverage. Each approach has its own set of benefits and is designed to test (cover) every part of software code. With Statement and Branch coverage, you can often obtain 80-90 percent code coverage, which is sufficient.

- Statement Coverage
- Decision Coverage
- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage

- Control flow testing
- Data flow testing

35. What is Experience-based testing techniques?

Discovery, research, and learning are all part of experience-based testing. The tester is continually studying and analysing the product and applying his abilities, traits, and experience to build test techniques and test cases in order to carry out the necessary testing. Various strategies for testing based on experience include:

- Exploratory Testing
- Error Guessing
- Random Testing

1. **Random Testing**: This is a black box testing technique in which all test inputs are produced at random (sometimes with the use of a tool) and then used for testing.
2. **Exploratory Testing**: This is a hands-on approach in which testers conduct as many tests as possible with as little planning as possible.
3. **Error Guessing**: This is a Software Testing technique for estimating the likelihood of an error occurring in the code. The major goal of this technique is to make educated guesses about potential defects in regions where formal testing isn't feasible.

36. What is a top-down and bottom-up approach in testing?

Top-down – In top-down Testing is done from the top to the bottom. To put it another way, high-level modules are evaluated first, followed by low-level modules. Finally, the low-level modules are merged into a high-level state to ensure that the framework performs as planned.

Advantages:

– Helpful if severe faults appear near the end of the programme.

– Once the I/O functions have been added, it is much easier to express test scenarios.

– The Early Skeletal Program promotes morale by allowing demonstrations.

Drawbacks:

– Stub modules must be created

– Stub Modules are frequently more difficult than they appear.

– It can be difficult to describe test scenarios in stubs before the I/O routines are added.

– Creating test conditions may be impossible or extremely complex.

– Examining test results is more challenging.

– Encourages the idea that design and testing can be combined.

– Induces one to postpone the completion of particular module testing.

Bottom-up — In Bottom-up Testing takes place at all levels, from the ground up. First, the lowest-level modules are examined, followed by high-level state modules. Finally, the high-level state modules are coordinated at a low level to ensure that the framework fills in as expected.

Advantages:

– Useful if severe faults appear near the end of the programme.

– Creating test conditions is less difficult.

– It is simpler to observe test outcomes.

Drawbacks:

– Driver Modules need to be created.

– The programme does not exist as a whole until the final module is added.

37. What is the difference between smoke testing and sanity testing?

Features	**Smoke Testing**	**Sanity Testing**
System Builds	Initial builds of software products are subjected to testing.	Builds that have passed smoke tests an regression tests are subjected to testing
Motive of Testing	To test the new build's stability before putting it through more rigorous testing	To assess the rationale and originality software builds' functionalities
Subset of?	Is a type of acceptability testing that is a subset of acceptance testing?	Is a type of regression testing that is a s of regression testing?
Documentation	Work includes documentation and programming.	Doesn't put a lot of emphasis on documentation.

Test Coverage	To integrate all of the essential capabilities without diving too deep, we took a shallow and broad approach.	An strategy that is both narrow and dee comprising extensive testing of functionalities and features.
Performed By?	Developers or testers are in charge of the execution.	Testers are in charge of the execution.

Key points about smoke testing and sanity testing:

Smoke testing is carried out at the start of the project to check for the most basic functions. Sanity testing, on the other hand, thoroughly checks software builds.

Smoke testing requires a written set of tests or automated tests for documentation, whereas sanity testing does not.

The smoke testing method is shallow and broad, which means it tests every build but does not go to extremes. Sanity testing, on the other hand, employs a focused and deep method in which a single build is extensively tested.

The smoke testing method is shallow and broad, which means it tests every build but does not go to extremes. Sanity testing, on the other hand, employs a focused and deep method in which a single build is extensively tested.

The fundamental goal of smoke testing is to swiftly cover every aspect of the software. Sanity testing, on the other hand, focuses on the functionality of each software module.

The developer is in charge of the smoke test, whereas the tester is in charge of the sanity test.

In a document verification procedure, smoke testing is similar to counting the number of documents. Sanity testing, on the other hand, entails a thorough examination of a single document.

38. What is the difference between static testing and dynamic testing?

Static testing	**Dynamic testing**
Static Testing is a white box testing technique that entails browsing records in order to identify flaws in the very early stages of the SDLC.	Dynamic testing is performed at the end of the software development lifecycle and involves the execution of code. verifies and approves the output to ensure that the desired outcomes are achieved.

Static Testing is used during the verification process.	During the validation step, dynamic testing begins.
Before deploying the code, static testing is carried out.	After the code has been deployed, dynamic testing is carri out.
In this form of testing, code error detection and programme execution are not a problem.	For dynamic testing, code execution is required.

39. How will you determine when to stop testing?

It can be tough to know when to stop testing. Many modern software applications are so complicated and run in such an interconnected environment that thorough testing is impossible. The following are some frequent criteria to consider when considering when to end testing:

- Deadlines are important (release deadlines, testing deadlines, etc.)
- Completed test cases with a particular percentage of them passing
- When the test budget runs out
- When the coverage of code, functionality, or requirements reaches a certain point, it is said to be complete.
- When the bug rate falls below a specific threshold,
- When the beta or alpha testing phase is over

To determine when to end testing, use the following scientific methods:

1) Decision based on the number of test cases that pass or fail:

a) Before the test execution cycle, prepare a predetermined number of test cases.

b) All test cases must be completed. During each testing cycle.

c) When all test cases are passed, the testing procedure is terminated.

d) Alternatively, if the proportion of failure in the previous testing cycle is exceedingly low, testing can be terminated.

2) Metric-driven decision:

a) Mean Time Between Failure (MTBF): this is calculated by calculating the average operational time before the system fails.

b) Coverage metrics: by keeping track of how many instructions were executed during tests.

c) Defect density: measuring the number of defects per 1000 lines of code or the number of open bugs, as well as their severity levels.

40. What if the software is so buggy it can't really be tested at all?

Frequently, testers come across a bug that cannot be fixed. In such cases, the best course of action is for testers to go through the process of reporting any flaws or blocking-type issues that arise, with a concentration on critical bugs. Because this type of issue might result in serious issues such as insufficient unit or integration testing, poor design, wrong build or release methods, and so on, management should be contacted and given documentation as proof of the problem.

41. How you test a product if the requirements are yet to freeze?

It's possible that a requirement stack for a product isn't available. It could take a lot of work to figure out if an app has a lot of unexpected functionality, and it could suggest deeper issues with the software development process. If a feature isn't required for the application's goal, it should be eliminated. Create a test plan based on the assumptions you've made about the product if all else fails. However, make sure that all of your assumptions are adequately documented in the test plan.

42. What if an organization is growing so fast that fixed testing processes are impossible? What to do in such situations?

This is a very prevalent issue in the software industry, especially with the new technologies that are being used in product development. In this case, there is no simple answer; however, you could:

• People who are good at what they do should be hired.

• Quality issues should be 'ruthlessly prioritised' by management, with a constant focus on the client.

• Everyone in the company should understand what 'quality' means to the customer.

43. How do you know the code has met specifications?

Code that works, is bug-free, and is understandable and manageable is considered "good code." Most organisations have coding 'standards' that all developers are expected to follow, but everyone has their own opinion on what is best, as well as how many restrictions are too many or too few. There are many methods available, such as a traceability matrix, to guarantee that requirements are linked to test cases. And when all of the test cases pass, that means the code satisfies the requirement.

44. What are the cases when you'll consider choosing automated testing over manual testing?

When the following scenarios apply, automated testing should be preferred over manual testing:

- When testing must be run on a regular basis
- Repetitive steps are included in the tests.
- When you have a limited amount of time to complete the testing phase, tests must be conducted in a regular runtime environment.
- When a large amount of code needs to be tested repeatedly
- Every execution necessitates a report.

45. What is 'configuration management'?

Every well-functioning company has a "master plan" that outlines how it will operate and complete tasks. It's the same with software development and testing. SCM is a collection of processes, policies, and tools for organising, controlling, coordinating, and tracking:

- code
- documentation
- problems
- change requests
- designs and tools
- compilers and libraries

46. Is it true that we can do system testing at any stage?

In system testing, all of the software's components are tested as a whole to guarantee that the final product fits the required requirements. As a result, no. Only until all of the units are in place and performing properly can the system testing begin. Prior to the UAT, system testing is frequently performed (User Acceptance Testing).

47. What are some best practices that you should follow when writing test cases?

- The following are some guidelines to follow when writing test cases:
- Prioritize which test cases to build based on your application's risk considerations and project timeframes.
- Keep the 80/20 rule in mind. To get the best coverage, you should have 20% of your tests cover 80% of your application.

Instead of trying to test all of your instances at once, improvise as you go.

- Make a list of all of your test cases and categorise them according to business scenarios and functionality.
- Ascertain that test cases are modular and that test case steps are as detailed as possible.
- Write test cases in a way that others can easily understand and alter them if necessary.
- Always keep end-user requirements in mind because, at the end of the day, the product is developed for the client.
- To maintain a solid release cycle, actively use a test management solution.
- Keep an eye on your test cases on a frequent basis. Remove irrelevant and duplicate test cases and write unique test cases.

48. Why is it that the boundary value analysis provides good test cases?

The fact that a greater number of errors occur at the boundaries rather than in the middle of the input domain for a test is one of the reasons why boundary value analysis produces effective test cases.

Test cases for the boundary value analysis technique are designed to include values at the edges. It's called 'Positive testing' if the input is within the boundary value. Negative testing happens when the input value is outside of the boundary value. Maximum, minimum, inside or outside edge, normal values, or error values are all included.

Assume you're looking for a number input box that takes numbers ranging from '01 to 10′.

We may divide test scenarios into three categories using boundary value analysis:

Test scenarios with test data that is identical to the input boundaries: 1 and ten (in this case)

Values just below the input domains' extreme edges: 0 and 9 are the two most common numbers. Test data with values right above the input domains' extreme edges: 11 and 2

As a result, the boundary values would be 0, 1, 2 and 9, 10, 11 respectively.

49. Why is it impossible to test a program thoroughly or in other terms 100% bug-free?

It is impossible to create a software product that is bug-free 100 percent of the time. You can simply minimise a computer program's or system's error, flaw, failure, or fault that causes it to generate an inaccurate or unexpected outcome.

Here are the two main reasons why testing a programme completely is impossible.

Software specifications are usually subjective, leading to a range of interpretations.

To test a software programme, it may be necessary to use too many inputs, outputs, and path combinations.

50. Can automation testing replace manual testing?

Manual testing is not replaced by automation testing. You can't automate everything, no matter how good your automated tests are. Manual tests are useful in software development because they can be used in situations where automation isn't possible.

Both automated and manual testing have advantages and disadvantages. Manual testing allows us to have a better understanding of the problem and explore different test angles with greater freedom.

Automated testing saves time in the long run by conducting a high number of surface-level tests in a short period of time.

Now, let's have a quick discussion about manual testing in general.

The following are some of the advantages of manual testing:

- For products with a short life cycle, this is the best option.
- It helps you save time, money, and resources.
- Ascertain that the product is free of errors.
- Ad hoc testing, exploratory testing, and usability testing are all possible with this tool.
- To make simple adjustments, there's no need to alter the complete code.
- Obtain precise feedback on the user interface
- Ability to better handle complex use case conditions
- GUI testing can be done with precision.
- Exceptionally dependable
- Improve user friendliness
- New testers will find it simple to pick up

The following are some of the disadvantages of manual testing:

- Not recommended for time-sensitive projects or large enterprises.
- Human errors and blunders are more likely to occur.
- Less efficient because there is no option to document the testing procedure.
- Less Dependable
- Testing for regression takes a long time.
- Doesn't cover everything there is to know about testing.
- Manual load testing and performance testing are possible.
- In the long term, the process is more costly.

Process of manual testing

- Planning and control
- Analysis and Design
- Implementation and Execution
- Evaluating exit criteria and Reporting
- Test closure activities

We reached the end of Manual Testing Interview Questions. Hope you find this helpful. All the best guys and stay tuned for more informative content like this.