Software Testing

Top 50 Manual Testing Interview Questions you Need to know In 2021

Testing is crucial to the success of any software product in this competitive world. Manual tests play a pivotal role in software development and come in handy whenever you have a case where you cannot use automated tests. Hence, there is still a lot of demand for people with skills relevant to manual testing. This Manual Testing Interview Questions article is the perfect guide for you to master software testing.

Let's begin by taking a look at the most frequently asked Manual Testing Interview Questions.

For better understanding, I have divided the rest of the Manual Testing Interview Questions into the following sections:

Basic Manual Testing Interview Questions

Q1. How does quality control differ from quality assurance?

Quality Control vs Quality Assurance	
Quality Control	**Quality Assurance**
Quality control is a product-oriented approach of running a program to determine if it has any defects, as well as making sure that the software meets all of the requirements put forth by the stakeholders	Quality assurance is a process-oriented approach that focuses on making sure that the methods, techniques, and processes used to create quality deliverables are applied correctly.

Q2. What is Software Testing?

Software Testing is a process used to identify the correctness, completeness, and quality of developed software. It includes a series of activities conducted with the intent of finding errors in software so that it could be corrected before the product is released to the market.

Q3. Why is Software Testing Required?

Software testing is a mandatory process that guarantees that the software product is safe and good enough to be released to the market. Here are some compelling reasons to prove testing is needed:

- It points out the defects and errors that were made during the development phases.
- Reduces the coding cycles by identifying issues at the initial stage of the development.
- Ensures that software application requires lower maintenance cost and results in more accurate, consistent and reliable results.
- Testing ensures that the customer finds the organization reliable and their satisfaction in the application is maintained.
- Makes sure that software is bug-free and the quality of the product meets the market standard.
- Ensures that the application doesn't result in any failures.

Q4. What are the two main categories of software testing?

Software testing is a huge domain but it can be broadly categorized into two areas such as :

- **Manual Testing** – This is the oldest type of software testing where the testers manually execute test cases without using any test automation tools. It means the software application is tested manually by QA testers.

- **Automation Testing** – This is the process of using the assistance of tools, scripts, and software to perform test cases by repeating pre-defined actions. Test Automation focuses on replacing manual human activity with systems or devices that enhance efficiency.

Become certified in automation testing get Selenium Online training!

Q5. What is quality control? Is it similar to Quality Assurance?

Quality control is a product-oriented approach of running a program to determine if it has any defects, as well as making sure that the software meets all of the requirements put forth by the stakeholders.

Q6. What different types of manual testing are there?

Different types of manual testing are;

-
  - Black Box Testing
  - White Box Testing
  - Unit Testing
  - System Testing
  - Integration Testing
  - Acceptance Testing

Q7. Explain the difference between alpha testing and beta testing.

- **Alpha Testing** – It is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is a type of user acceptance testing.
- **Beta Testing** – It is performed by real users of the software application in a real environment. Beta Testing is also a type of user acceptance testing.

Q8. What are the different levels of manual testing?

Four levels of manual testing are:

- **Unit testing** – It is a way of testing the smallest piece of code referred to as a **unit** that can be logically isolated in a system. It is mainly focused on the functional correctness of the standalone module.
- **Integration Testing** – It is a level of software testing where individual units are combined and tested to verify if they are working as they intend to when integrated. The main aim here is to test the interface between the modules.
- **System Testing** – In system testing all the components of the software are tested as a whole in order to ensure that the overall product meets the requirements specified. There are dozens of types of system testing, including usability testing, regression testing, and functional testing.

User Acceptance Testing – The final level, acceptance testing, or UAT (user acceptance testing), determines whether or not the software is ready to be released.

Q9. What is a testbed in manual testing?

The testbed is an environment configured for testing. It is an environment used for testing an application, including the hardware as well as any software needed to run the program to be tested. It consists of hardware, software, network configuration, an application under test, other related software.

Q10. Explain the procedure for manual testing?

The manual testing process comprises the following steps:

- Planning and Control
- Analysis and Design
- Implementation and Execution
- Evaluating exit criteria and Reporting
- Test Closure activities

In case you are facing any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

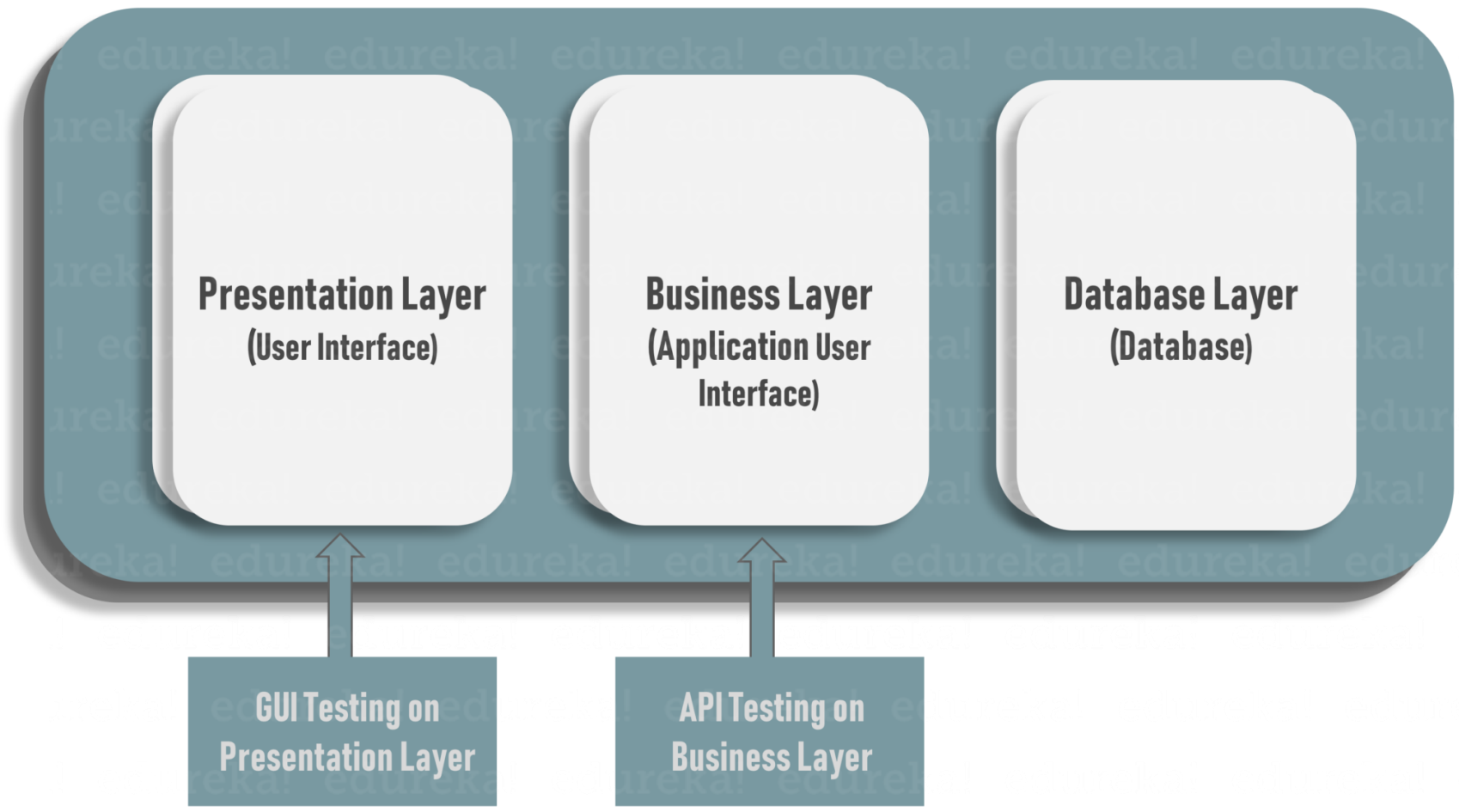
Q11. What is the test case?

A **test case** is a document that has a set of conditions or actions that are performed on the software application in order to verify the expected functionality of the feature.

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

Q12. What is API testing?

API testing is a type of software testing where application programming interfaces (APIs) are tested to determine if they meet expectations for functionality, reliability, performance, and security. In simple terms, API testing is intended to reveal bugs, inconsistencies or deviations from the expected behavior of an API.



Commonly, applications have three separate layers:

- Presentation Layer or user interface
- Business Layer or application user interface for business logic processing
- Database Layer for modeling and manipulating data

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

Q13. What's the difference between verification and validation in testing?

Verification	Validation
It is a static analysis technique. Here, testing is done without executing the code. Examples include – Reviews, Inspection, and walkthrough.	It is a dynamic analysis technique where testing is done by executing the code. Examples include functional and non-functional testing techniques.

Q14. What's the difference between a bug and a defect?

A bug is a just fault in the software that's detected during testing time. A defect is a variance between expected results and actual results, detected by the developer after the product goes live.

Q15.What are the advantages of manual testing?

Merits of manual testing are:

- It is a cheaper way of testing when compared to automated testing
- Analysis of product from the point of view of the end-user is possible only with manual testing
- GUI testing can be done more accurately with the help of manual testing as visual accessibility and preferences are difficult to automate
- East to learn for new people who have just entered into testing
- It is highly suitable for short-term projects when test-scripts are not going to be repeated and reused for thousands of times
- Best suited when the project is at the early stages of its development
- Highly reliable, since automated tests can contain errors and missed bugs

Q16.What are the drawbacks of manual testing?

De-merits of manual testing are:

- Highly susceptible to human error and are risky
- Test types like load testing and performance testing are not possible manually
- Regression tests are really time-consuming if they are done manually
- Scope of manual testing is very limited when compared to automation testing
- Not suitable in very large organizations and time-bounded projects
- The cost adds up, so, it's more expensive to test manually in the long run

Q17. What's the role of documentation in Manual Testing?

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

Documenting the test cases will facilitate you to estimate the testing effort you will need along with test coverage and tracking and tracing requirement. Some commonly applied documentation artifacts associated with software testing are:

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

With this, we have completed basic questions based on manual testing. In the next part of this *Manual Testing Interview Questions article,* let's discuss advanced level questions related to manual testing.

Advanced Level Manual Testing Interview Questions

Q18. What is the difference between manual testing and automation testing?

Manual Testing	Automation Testing
In manual testing, the accuracy, and reliability of test cases are low, as manual tests are more prone to human error.	Automated testing, on the other hand, is more reliable as tools and scripts are used to perform tests.
The time required for manual testing is high as human resources perform all the tasks.	The time required is comparatively low as software tool execute the tests
In manual testing investment cost is low, but Return of Investment(ROI) is low as well.	In automation testing investment cost and Return of Investment, both are high.
Manual testing is preferred when the test cases are run once or twice. Also suitable for Exploratory, Usability and Adhoc Testing.	You can use test automation for Regression Testing, Performance Testing, Load Testing or highly repeatable functional test cases
Allows for human observation to find out any glitches. Therefore manual testing helps in improving the customer experience.	As there is no human observation involved, there is no guarantee of positive customer experience.

Q19. When should you opt for manual testing over automation testing?

There are a lot of cases when manual testing is best suited over automation testing, like:

- **Short-time projects:** Automated tests are aimed at saving time and resources yet it takes time and resources to design and maintain them. For example, if you are building a small promotional website, it can be much more efficient to rely on manual testing.
- **Ad-hoc Testing:** In ad-hoc testing, there is no specific approach. Ad-hoc testing is a totally unplanned method of testing where the understanding and insight of the tester is the only important factor. This can be achieved using manual testing.
- **Exploratory Test:** This type of testing requires the tester's knowledge, experience, analytical, logical skills, creativity, and intuition. So human involvement is important in exploratory testing.
- **Usability Testing:** When performing usability testing, the tester needs to measure how user-friendly, efficient, or convenient the software or product is for the end-users. Human observation is the most important factor, so manual testing sounds seems more appropriate.

Q20. What are the phases involved in Software Testing Life Cycle?

The different phases involved in the software testing life cycle are:

Phases	Explanation

Requirement Analysis	QA team understands the requirement in terms of what we will testing & figure out the testable requirements.
Test Planning	In this phase, the test strategy is defined. Objective & the scope of the project is determined.
Test Case Development	Here, detailed test cases are defined and developed. The testing team also prepares the test data for testing.
Test Environment Setup	It is a setup of software and hardware for the testing teams to execute test cases.
Test Execution	It is the process of executing the code and comparing the expected and actual results.
Test Cycle Closure	It involves calling out the testing team member meeting & evaluating cycle completion criteria based on test coverage, quality, cost, time, critical business objectives, and software.

In case you are facing any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

Q21. What is the difference between a bug, a defect and an error?

Bug – A bug is a fault in the software that's detected during testing time. They occur because of some coding error and leads a program to malfunction. They may also lead to a functional issue in the product. These are fatal errors that could block a functionality, results in a crash, or cause performance bottlenecks

Defect – A defect is a variance between expected results and actual results, detected by the developer after the product goes live. The defect is an error found AFTER the application goes into production. In simple terms, it refers to several troubles with the software products, with its external behavior, or with its internal features.

Error – An error is a mistake, misunderstanding, or misconception, on the part of a software developer. The category of developers includes software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a design notation, or a programmer might type a variable name incorrectly – leads to an error. An error normally arises in software, it leads to a change the functionality of the program.

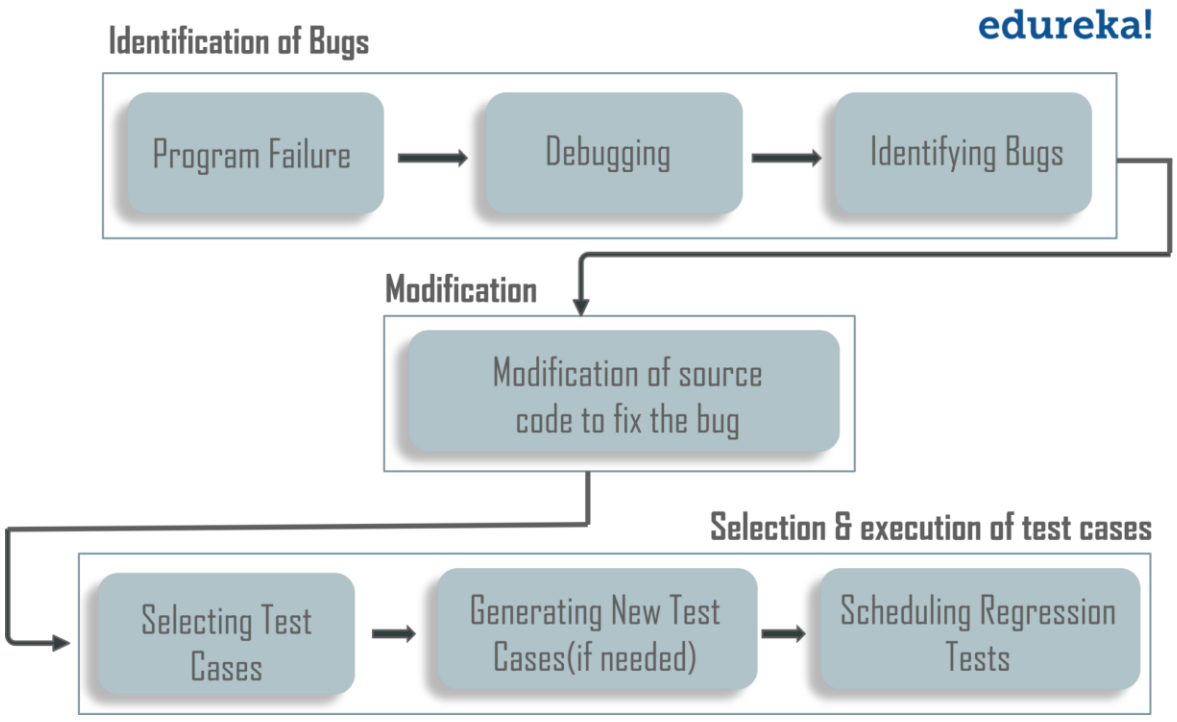
Q22. What makes a good test engineer?

A **software test engineer** is a professional who determines how to create a process that would best *test* a particular product in the software industry.

- A good test engineer should have a 'test to break' attitude, an ability to take the point of view of the customer
- Strong desire for quality and attention to minute details
- Tact and diplomacy to maintain a cooperative relationship with developers
- Ability to communicate with both technical (developers) and non-technical (customers, management) people
- Prior experience in the software development industry is always a plus
- Ability to judge the situations and make important decisions to test high-risk areas of an application when time is limited

Q23.What is regression testing? When to apply it?

"Testing of a previously tested program to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made is called Regression Testing."



A regression test is a system-wide test whose main purpose is to ensure that a small change in one part of the system does not break existing functionality elsewhere in the system. It is recommended to perform regression testing on the occurrence of the following events:

- When new functionalities are added
- In case of change requirements

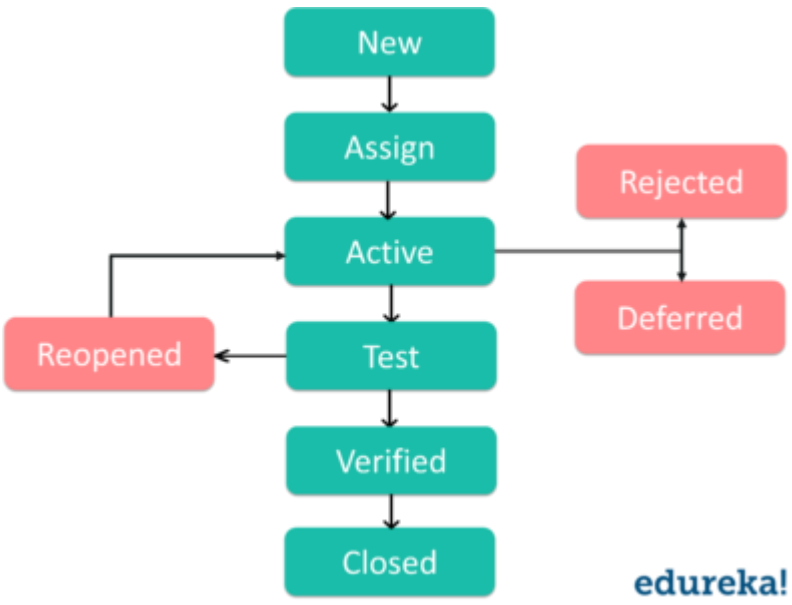
- When there is a defect fix
- When there are performance issues
- In case of environment changes
- When there is a patch fix

Q24. What is the difference between system testing and integration testing?

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

Q25. Explain the defect life cycle.

A **defect life cycle** is a process 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 closed, after ensuring it's not reproduced. Bug or defect life cycle includes the steps as shown in the below figure.



If you wish to learn in-depth about Bug Life Cycle then you can refer this article on Software Testing Tutorial.

Q26. What is the test harness?

A test harness is the gathering of software and test information arranged to test a program unit by running it under changing conditions like stress, load, data-driven, and monitoring its behavior and outputs. Test Harness contains two main parts:

– A Test Execution Engine
– Test script repository

Q27. What is test closure?

Test Closure is a document which gives a summary of all the tests conducted during the software development life cycle and also gives a detailed analysis of the bugs removed and errors found. This memo contains the aggregate no. of experiments, 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 forth.

Q28. What is the difference between Positive and Negative Testing?

Positive Testing	Negative Testing
Positive testing determines that your application works as expected. If an error is encountered during positive testing, the test fails	**Negative testing** ensures that your application can gracefully handle invalid input or unexpected user behavior
In this testing, tester always check for an only valid set of values	Testers apply as much creativity as possible and validating the application against invalid data

Q29. Define what is a critical bug.

A critical bug is a bug that has got the tendency to affect a majority of the functionality of the given application. It means a large piece of functionality or major system component is completely broken and there is no workaround to move further. Application cannot be distributed to the end client unless the critical bug is addressed.

Q30. What is the pesticide paradox? How to overcome it?

According to *pesticide paradox*, if the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs. Developers will be extra careful in those places where testers found more defects and might not look into other areas. **Methods to prevent pesticide paradox:**

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

Using these methods, it's possible to find more defects in the area where defect numbers dropped.

In case you are facing any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

Q31. What is Defect Cascading in Software Testing?

Defect Cascading is the process of triggering other defects in the application. When a defect goes unnoticed while testing, it invokes other defects. As a result, multiple defects crop up in the later stages of development. If defect cascading continues to affect other features in the application, identifying the affected feature becomes challenging. You may make different test cases to solve this issue, even then it is difficult and time-consuming.

Q32. What is the term 'quality' mean when testing?

In general, quality software is reasonably bug-free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable. But again 'quality' is a subjective term. It will depend on who the 'customer' is and their overall influence in the scheme of things. For example, each type of 'customer' will have their own slant on 'quality' – the accounting department might define quality in terms of profits while an end-user might define quality as user-friendly and bug-free.

Q33. What is black box testing, and what are the various techniques?

Black-Box Testing, also known as specification-based testing, analyses the functionality of a software/application without knowing much about the internal structure/design of the item. The purpose of this testing is to check the functionality of the system as a whole to make sure that it works correctly and meets user demands. Various black-box testing techniques are:

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Based Technique
- Cause-effect Graphing
- Use Case Testing

Q34. What is white box testing, and what are the various techniques?

White-Box Testing also known as structure-based testing, requires a profound knowledge of the code as it includes testing of some structural part of the application. The purpose of this testing is to enhance security, check the flow of inputs/outputs through application and to improve design and usability. Various white-box testing techniques are:

- Statement Coverage
- Decision Coverage
- Condition Coverage
- Multiple Condition Coverage

Q35. What are the Experience-based testing techniques?

Experienced-based testing is all about discovery, investigation, and learning. The tester constantly studies and analyzes the product and accordingly applies his skills, traits, and experience to develop test strategies and test cases to perform necessary testing. Various experience-based testing techniques are:

- Exploratory Testing
- Error Guessing

Q36.What is a top-down and bottom-up approach in testing?

Top-Down – Testing happens from top to bottom. That is, high-level modules are tested first, and after that low-level modules. Lastly, the low-level modules are incorporated into a high-level state to guarantee the framework is working as it is expected to.

Bottom-Up – Testing happens from base levels to high-up levels. The lowest level modules are tested first and afterward high-level state modules. Lastly, the high-level state modules are coordinated to a low level to guarantee the framework is filling in as it has been proposed to.

Q37. What is the difference between smoke testing and sanity testing?

Features	Smoke Testing	Sanity Testing
System Builds	Tests are executed on initial builds of software product	Tests are done on builds that have passed smoke tests & rounds of regression tests
Motive of Testing	To measure the stability of the newly created build to face off more rigorous testing	To evaluate rationality & originality of the functionalities of software builds
Subset of?	Is a subset of acceptance testing	Is a subset of regression testing
Documentation	Involves documentation and scripting work	Doesn't emphasize any sort of documentation
Test Coverage	Shallow & wide approach to include all the major functionalities without going too deep	Narrow & deep approach involving detailed testing of functionalities and features
Performed By?	Executed by developers or testers	Executed by testers

Q38. What is the difference between static testing and dynamic testing?

Static Testing	Dynamic Testing
Static Testing is a white box testing technique, it includes the process of exploring the records to recognize the imperfections in the very early stages of SDLC.	Dynamic testing includes the process of execution of code and is done at the later stage of the software development lifecycle. It validates and approves the output with the expected results.
Static Testing is implemented at the verification stage.	Dynamic testing starts during the validation stage.
Static testing is performed before the code deployment.	Dynamic testing is performed after the code deployment
The code error detection and execution of the program is not a concern in this type of testing.	Execution of code is necessary for dynamic testing.

With this, we have completed theory questions. In the next part of this *Manual Testing Interview Questions article,* let's discuss some real-world scenario-based questions.

Real-World Based Manual Testing Interview Questions

Q39. How will you determine when to stop testing?

Deciding when to stop testing can be quite difficult. Many modern software applications are so complex and run in such an interdependent environment, that complete testing can never be done. Some common factors in deciding when to stop testing are:

- Deadlines (release deadlines, testing deadlines, etc.)
- Test cases completed with certain percentage passed
- When the test budget is depleted
- Coverage of code or functionality or requirements reaches a specified point
- Bug rate falls below a certain level
- When Beta or alpha testing period ends

Q40. What if the software is so buggy it can't really be tested at all?

Often testers encounter a bug that can't be resolved at all. In such situations, the best bet is for testers to go through the process of reporting whatever bugs or blocking-type problems initially show up, with the focus being on critical bugs. Since this type of problem can cause severe problems such as insufficient unit testing or insufficient integration testing, poor design, improper build or release procedures, etc managers should be notified and provided with some documentation as evidence of the problem.

In case you are facing any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

Q41. How you test a product if the requirements are yet to freeze?

It's possible that a requirement stack is not available for a piece of product. It might take serious effort to determine if an application has significant unexpected functionality, and it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed. Else, create a test plan based on the assumptions made about the product. But make sure you get all assumptions well documented in the test plan.

Q42. What if an organization is growing so fast that fixed testing processes are impossible? What to do in such situations?

This is a very common problem in the software industry, especially considering the new technologies that are being incorporated when developing the product. There is no easy solution in this situation, you could:

• Hire good and skilled people
• Management should ‘ruthlessly prioritize’ quality issues and maintain focus on the customer
• Everyone in the organization should be clear on what ‘quality’ means to the end-user

Q43. How do you know the code has met specifications?

‘Good code’ is code that works, that is bug-free, and is readable and maintainable. Most organizations have coding ‘standards’ that all developers are supposed to adhere to, but everyone has different ideas about what’s best, or what is too many or too few rules. There are a lot of tools like traceability matrix which ensures the requirements are mapped to the test cases. And when the execution of all test cases finishes with success, it indicates that the code has met the requirement.

Q44. What are the cases when you’ll consider to choose automated testing over manual testing?

Automated testing can be considered over manual testing during the following situations:

- When tests require periodic execution
- Tests include repetitive steps
- Tests need to be executed in a standard runtime environment
- When you have less time to complete the testing phase
- When there is a lot of code that needs to be repeatedly tested
- Reports are required for every execution

Q45. What is ‘configuration management’?

Every high-functioning organization has a “master plan” that details how they are supposed to operate and accomplish tasks. Software development and testing are no different. Software configuration management (SCM) is a set of processes, policies, and tools that organize, control, coordinate, and track:

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

Q46. Is it true that we can do system testing at any stage?

In system testing, all the components of the software are tested as a whole in order to ensure that the overall product meets the requirements specified. So, no. The system testing must start only if all units are in place and are working properly. System testing usually happens before the UAT (User Acceptance Testing).

Q47. What are some best practices that you should follow when writing test cases?

Few guidelines that you need to follow while writing test cases are:

- Prioritize which test cases to write based on the project timelines and the risk factors of your application.
- Remember the 80/20 rule. To achieve the best coverage, 20% of your tests should cover 80% of your application.
- Don’t try to test cases in one attempt instead improvise them as you progress.
- List down your test cases and classify them based on business scenarios and functionality.
- Make sure test cases are modular and test case steps are as granular as possible.
- Write test cases in such a way that others can understand them easily & modify if required.
- Always keep end-users’ requirements in the back of your mind because ultimately the software designed is for the customer
- Actively use a test management tool to manage stable release cycle.
- Monitor your test cases regularly. Write unique test cases and remove irrelevant & duplicate test cases.

Q48. Why is it that the boundary value analysis provides good test cases?

The reason why boundary value analysis provides good test cases is that usually, a greater number of errors occur at the boundaries rather than in the center of the input domain for a test.

In boundary value analysis technique test cases are designed to include values at the boundaries. If the input is within the boundary value, it is considered ‘Positive testing.’ If the input is outside of the boundary value, it is considered ‘Negative testing.’ It includes maximum, minimum, inside or outside edge, typical values or error values.

Let’s suppose you are testing for an input box that accepts numbers from ’01 to 10'.

Using the boundary value analysis we can define three classes of test cases:

- Test cases with test data exactly as the input boundaries of input: 1 and 10 (in this case)
- Values just below the extreme edges of input domains: 0 and 9
- Test data with values just above the extreme edges of input domains: 2 and 11

So the boundary values would be 0, 1, 2 and 9, 10, 11.

Q49.Why is it impossible to test a program thoroughly or in other terms 100% bug-free?

It is impossible to build a software product which is 100% bug-free. You can just minimize the error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result.

Here are the two principal reasons that make it impossible to test a program entirely.

- Software specifications can be subjective and can lead to different interpretations.
- A software program might require too many inputs, too many outputs, and too many path combinations to test.

Q50. Can automation testing replace manual testing?

Automation testing isn't a replacement for manual testing. No matter how good automated tests are, you cannot automate everything. Manual tests play an important role in software development and come in handy whenever you have a case where you cannot use automation. Automated and manual testing each have their own strengths and weaknesses. Manual testing helps us to understand the entire problem and explore other angles of tests with more flexibility. On the other hand, automated testing helps save time in the long run by accomplishing a large number of surface-level tests in a short time.