Manual Testing Interview Questions

Click here to view the live version of the page

Top Answers to Manual Testing Interview Questions

Before the launch of any product or software, testing is a must. You can use automation testing in most cases but not for all of them. This is where manual testing comes in and it plays an important role in the field of software development. Our blog on manual testing interview questions and answers will guide you to master the field with the help of a carefully collated set of interview questions on manual testing.

Mentioned below are the different categories into which the following manual testing interview questions have been classified:

Basic Manual Testing Interview Questions for Freshers

Intermediate Manual Testing Interview Questions

Advanced Manual Testing Interview Questions for Experienced

Manual Testing Interview Questions For 2 Years Experience

Manual Testing Interview Questions For 3 Years Experience

Manual Testing Interview Questions For 4 Years Experience

Manual Testing Interview Questions For 5 Years Experience

Manual Testing Scenario-Based Interview Questions

Salary in Manual Testing based on Skills

Manual Testing Job Trends in 2024

Job Opportunities in Manual Testing

Roles and Responsibilities in Manual Testing

Conclusion

Did you know?

- Two-thirds of software development businesses use testing in a 75:25 (manual: automated) ratio or 50:50
- 49.8% of manual testers are women, while 50.2% are men

Basic Manual Testing Interview Questions for Freshers

1. What do you understand by software testing?

Software testing is a validation process that confirms that a system works as per the business requirements. It qualifies a system on various aspects such as usability, accuracy, completeness, efficiency, etc. ANSI/IEEE 1059 is the global standard that defines the basic principles of testing.

2. When should you stop the testing process?

The testing activity ends when the testing team completes the following milestones.

Test case execution

The successful completion of a full test cycle after the final bug fix marks the end of the testing phase.

Testing deadline

The end date of the validation stage also declares the closure of the validation if no critical or high-priority defects remain in the system.

Code Coverage(CC) ratio

It is the amount of code concealed via automated tests. If the team achieves the intended level of code coverage (CC) ratio, then it can choose to end the validation.

Mean Time Between Failure (MTBF) rate

Mean time between failure (MTBF) refers to the average amount of time that a device or product functions before failing. This unit of measurement includes only operational time between failures and does not include repair times, assuming the item is repaired and begins functioning again. MTBF figures are often used to project how likely it is for a single unit to fail within a certain period of time.

3. What do verification and validation mean in software testing?

Verification is a process that confirms that product development takes place as per the specifications and uses standard development procedures. The process comprises the following activities:

- Inspections
- Reviews
- Walk-throughs
- Demos

Validation is a means to confirm that the developed product doesn't have any bugs and works as expected. It comprises the following activities:

- Functional testing
- Non-functional testing

Preparing for a Job Interview! Check out our Top Software Testing Interview Questions.

4. What is static testing? When does it start and what does it cover?

Static testing is a white-box testing technique that directs developers to verify their code with the help of a checklist to find errors in it. Developers can start the static testing without actually finalizing the application or program. Static testing is more cost-effective than dynamic testing as it covers more areas than dynamic testing in a shorter time.

5. Define black-box testing.

It is a standard software testing approach that requires testers to assess the functionality of the software as per the business requirements. The software is treated as a black box and validated as per the end user's point of view.

Check out our blog on Selenium tutorial to gain in-depth insights on Selenium!

6. What is a test plan and what does it include?

A test plan stores all possible testing activities to ensure a quality product. It gathers data from the product description, requirement, and use case documents.

The test plan document includes the following:

- Testing objectives
- Test scope
- Testing the frame
- Environment
- Reason for testing
- Criteria for entrance and exit
- Deliverables
- Risk factors

7. What is meant by test coverage?

Test coverage is a quality metric to represent the amount (in percentage) of testing that has been completed. It is relevant for both functional and non-functional testing activities. This metric is used to add missing test cases.

Don't miss out Automation Testing Interview Questions. Crack your interviews with ease.

8. Is it possible to achieve 100% testing coverage? How would you ensure it?

It's considered impossible to perform 100% testing of any product. But, you can follow the below steps to come closer.

- Set a hard limit on the following factors:
    - Percentage of test cases passed
    - Number of bugs found
- Set a red flag if:
    - Test budget is depleted
    - Deadlines are breached
- Set a green flag if:
    - The entire functionality gets covered in test cases
    - All critical and major bugs must have a 'CLOSED' status

9. What are unit testing and integration testing?

Unit testing has many names such as module testing or component testing.

Many times, it is the developers who test individual units or modules to check if they are working correctly.

Whereas, integration testing validates how well two or more units of software interact with each other.

There are three ways to validate integration:

- Big Bang approach
- Top-down approach
- Bottom-up approach

10. Can we do system testing at any stage?

No. System testing should start only if all modules are in place and they work correctly. However, it should be performed before UAT (user acceptance testing).

11. Mention the different types of software testing.

Various types of Software Testing used by manual testers are as follows:

- Black Box Testing
- Regression testing
- Smoke testing
- Functional testing
- Exploratory Testing
- Integration Testing
- System Testing
- Graphical User Interface Testing
- User Acceptance Testing (UAT)
- Alpha and Beta testing
- Unit testing
- Integration testing
- Shakeout testing
- Performance Testing

12. What is the difference between a test driver and a test stub?

The test driver is a section of code that calls a software component under test. It is useful in testing that follows the bottom-up approach.

The test stub is a dummy program that integrates with an application to complete its functionality. It is relevant for testing that uses the top-down approach.

For example:

1. Let's assume a scenario where we have to test the interface between modules A and B and we have developed only Module A. Here, we can test module A if we have the real Module B or a dummy module for it. In this case, we call module B as the test stub.
2. Now, module B can't send or receive data directly from module A. In such a scenario, we have to move data from one module to another using some external features called test driver.

13. What is agile testing and why is it important?

Agile testing is a software testing process that evaluates software from the customers' point of view. It is favorable as it does not require the development team to complete coding to start QA. Instead, both coding and testing go hand in hand. However, it may require continuous customer interaction.

14. What do you know about data flow testing?

It is a white-box testing techniques.

Data flow testing focuses on the creation of test cases that encompass the control flow paths involving variable declarations and their utilization within modules. It expects test cases to have the following attributes:

1. The input to the module
2. The control flow path for testing
3. A pair of an appropriate variable definition and its use
4. The expected outcome of the test case

15. What is the purpose of the end-to-end testing?

End-to-end testing is a testing strategy to execute tests that cover every possible flow of an application from its start to finish. The objective of performing end-to-end tests is to discover software dependencies and to assert that the correct input is getting passed between various software modules and sub-systems.

Also, check out the difference between automation and manual testing.

16. Can you explain the importance of test cases in the manual testing process?

In testing, test cases play a role due to various reasons:

1. Guidance and Consistency: Test cases provide a framework that guides testers through defined steps and expected outcomes consistently. This approach promotes uniformity in testing procedures among testers and testing cycles.
2. Coverage and Validation: Test cases methodically cover a range of functionalities, user scenarios, and error conditions to ensure test coverage. By validating these aspects, it confirms that the software meets specified requirements and functions correctly under certain situations.
3. Repeatability and Reliability: documented test cases allow for the reproduction of test scenarios, ensuring consistent results. This reliability is essential for verifying fixes, conducting regression tests, and maintaining software quality in the long run.
4. Effectiveness: Test cases streamline the testing process, helping testers execute tests efficiently without guesswork. This streamlined approach allows testers to focus on identifying defects and concentrate their testing efforts on specific areas of the software.
5. Documentation and Communication: Test cases serve as a form of documentation by detailing test scenarios, anticipated results, and actual findings. This helps to promote communication, among testers,

developers, and stakeholders, ensuring that everyone is on the same page regarding testing goals and standards for assessing software quality.

17. What is regression testing, and why is it important?

Regression testing is vital for software stability and quality assurance. It ensures that recent code changes don't introduce bugs or break existing features. By preventing regression issues, it supports agile development practices and continuous integration.

18. How do you prioritize test cases when you have limited time for testing?

We can prioritize test cases when we have limited time for testing in several scenario, including:

1. Identify critical functionalities: Focus on testing essential software features critical for user satisfaction and core operations.
2. Assess risk impact: Prioritize test cases based on their potential impact on software stability, user experience, and business objectives.
3. Consider frequency of use: Test functionalities frequently used to ensure thorough evaluation of critical aspects.
4. Focus on high risk areas: Allocate more time in testing areas to defects, implemented features, or complex functionalities.
5. Engage with stakeholders: Talk to project participants to coordinate testing strategies with business objectives and confirm that priorities match the requirements.

19. Can you explain the concept of equivalence partitioning and give an example?

Equivalence partitioning optimizes testing by dividing a system input domain into classes, reducing the number of test cases while ensuring thorough coverage. Here's how it works:

1. Divide Input Domain: System inputs are grouped into partitions, assuming similar behavior within each partition.
2. Select Representative Values: Test cases are chosen to represent each partition, ensuring comprehensive testing.
3. Execute Test Cases: Testing with these values validates system behavior within each partition.
4. Identify Defects: Discrepancies are noted and addressed, with additional cases added as needed.

Example: A login page partitions usernames and passwords into valid, invalid, and blank entries, enabling efficient testing.

20. How do you ensure adequate test coverage in your testing process?

To ensure comprehensive test coverage in your testing process, consider the following strategies:

1. Thorough Requirement Analysis: Understand project requirements to identify all functionalities and scenarios requiring testing.
2. Risk-Focused Testing: Arrange testing according to the consequences and chances of failure for each feature.
3. Thorough Test Scenario Development: Develop test scenarios that address negative and boundary conditions.
4. Exploratory Testing: Perform tests to uncover issues.
5. Code Coverage Evaluation: Employ tools to track code execution during testing, guaranteeing that all code pathways are assessed.
6. Traceability Matrix: Map test cases to requirements for tracking and ensuring all requirements are tested.

7. Continuous Improvement: Regularly update test cases to reflect changes, ensuring evolving coverage over time.

Intermediate Manual Testing Interview Questions

21. The probability that a server-class application hosted on the cloud is up and running for six long months without crashing is 99.99 percent. To analyze this type of scenario, what test will you perform?

Reliability testing

22. What will you do when a bug turns up during testing?

When a bug occurs, we can follow the below steps.

- We can run more tests to make sure that the problem has a clear description.
- We can also run a few more tests to ensure that the same problem doesn't exist with different inputs.
- Once we are certain of the full scope of the bug, we can add details and report it.

23. Why is it impossible to test a program thoroughly?

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 may require too many inputs, outputs, and path combinations.

24. How do you test a product if the requirements are yet to be freezed?

If the required specifications are not available for a product, then a test plan can be created based on the assumptions made about the product. But all assumptions must be well-documented in the test plan.

25. If a product is in the production stage and one of its modules gets updated, then is it necessary to perform regression testing?

Yes, it is necessary to perform regression testing when a module of a product in the production stage gets updated. Regression testing helps ensure that the changes made to the updated module do not have unintended effects on other modules or the overall functionality of the product. By retesting the previously working functionalities, it helps identify any potential issues or regressions caused by the module update. This testing process helps maintain the quality and stability of the product throughout its lifecycle.

26. How will you overcome the challenges faced due to the unavailability of proper documentation for testing?

If standard documents like the system requirement specification or feature description document are not available, then QA may have to rely on the following references, if available.

- Screenshots

- A previous version of the application
- Wireframes

Another reliable way is to have discussions with the developer and the business analyst. It helps in solving the doubts, and it opens a channel for bringing clarity on the requirements. Also, the emails exchanged could be useful as a testing reference.

Smoke testing is yet another option that would help verify the main functionality of the application. It would reveal some very basic bugs in the application. If none of these work, then we can just test the application from our previous experiences.

27. Is there any difference between retesting and regression testing?

Differences between retesting and regression testing are as follows:

- We perform retesting to verify the defect fixes. But, regression testing assures that the bug fix does not break other parts of the application.
- Regression test cases verify the functionality of some or all modules.
- Regression testing ensures the re-execution of passed test cases. Whereas, retesting involves the execution of test cases that are in a failed state.
- Retesting has a higher priority over regression. But in some cases, both get executed in parallel.

28. What are the different types of functional testing?

Functional testing covers the following types of validation techniques:

- Unit testing
- Smoke testing
- UAT
- Sanity testing

- Interface testing
- Integration testing
- System testing
- Regression testing

29. What are functional test cases and non-functional test cases?

- Functional Test Cases: Functional test cases are designed to evaluate the functionality of a software system or application. These test cases focus on verifying whether the system performs its intended functions correctly and meets the specified functional requirements. Functional test cases typically involve validating inputs, testing different scenarios, and verifying expected outputs.
- Non-Functional Test Cases: Non-functional test cases, on the other hand, assess the non-functional aspects of a software system or application. These test cases evaluate performance, usability, reliability, security, scalability, and compatibility. Non-functional testing cases ensure the system meets the required quality standards and provides a satisfactory user experience.

30. What do you understand about STLC?

Software testing life cycle (STLC) proposes the test execution in a planned and systematic manner. In the STLC model, many activities occur to improve the quality of the product.

The STLC model lays down the following steps:

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup

5. Test Execution
6. Test Cycle Closure

31. In software testing, what does a fault mean?

A fault is a condition that makes the software fail to execute while performing the considered function.

32. Difference between bug, defect, and error.

A slip in coding is indicated as an error. The error spotted by a manual tester becomes a defect. The defect which the development team admits is known as a bug. If a built code misses on the requirements, then it is a functional failure.

33. How do severity and priority relate to each other?

Severity: It represents the gravity/depth of a bug. It describes the application point of view.

Priority: It specifies which bug should get fixed first. It defines the user's point of view.

Enroll now in Selenium course to learn more about selenium concepts!

34. List the different types of severity.

The criticality of a bug can be low, medium, or high depending on the context.

- User interface defects – Low
- Boundary-related defects – Medium
- Error handling defects – Medium
- Calculation defects – High
- Misinterpreted data – High
- Hardware failures – High
- Compatibility issues – High
- Control flow defects – High
- Load conditions – High

35. What is the purpose of a traceability matrix, and how do you create and maintain one?

The traceability matrix aligns requirements with test cases, ensuring thorough test coverage. To create and maintain it, identify project artifacts, establish traceability links, and update it continuously. Use the matrix for reporting and analysis to track test coverage effectively.

36. How do you handle test data management and ensure data integrity during testing?

Managing test data and ensuring its integrity is crucial for effective testing. Here's a concise approach:

1. Identify Relevant Data: Identify necessary test data for scenarios.
2. Generate Data: Use tools/scripts to generate diverse test data.
3. Protect Sensitive Information: Mask or anonymize PII to safeguard data.
4. Ensure Data Separation: Keep test and production data isolated.
5. Validate Data Integrity: Perform checks to validate data accuracy during testing.

37. Can you describe the defect life cycle and the different stages involved in defect management?

The defect life cycle outlines stages in defect management:

- Identification: The defect is logged.
- Assignment: A team member analyzes the defect.
- Open: The defect is confirmed.
- In Progress:The developer fixes the defect.
- Fixed: The defect is resolved.
- Retesting:The defect is ready for testing.
- Reopened: If necessary, the defect is revisited.
- Closed: Verified defect closure.

38. What techniques do you use to test database views and stored procedures using SQL queries? How do you ensure their correctness and efficiency?

To test database views and stored procedures effectively:

- Input Validation: Validate inputs and outputs for expected results.
- Functional Testing: Confirm views/procedures meet requirements.
- Boundary Testing: Test extreme input values for edge cases.
- Performance Testing: Assess query execution time and resource usage.
- Integration Testing: Verify compatibility with other components.

To ensure the correctness and efficiency of database views and stored procedures:

- Correctness: Validate inputs and outputs against requirements. Compare the results with expectations to detect discrepancies. Ensure compliance with business rules.
- Efficiency: Analyze SQL queries for performance issues. Optimize queries by adding indexes, rewriting SQL, or restructuring the database schema. Monitor query execution time and resource usage for improvements.

39. Describe the process you follow for creating and executing test cases in a manual testing environment.

Creating and executing test cases in a manual testing environment involves several key steps:

1. Requirement Analysis: Thoroughly understand project requirements.
2. Test Planning: Develop a comprehensive test plan.
3. Test Case Design: Create detailed test cases for various scenarios.
4. Test Data Preparation: Gather or create the necessary test data.
5. Test Environment Setup: Ensure the readiness of the testing environment.
6. Test Execution: Meticulously execute test cases and record results.
7. Defect Reporting: Document encountered defects with clear descriptions.
8. Defect Tracking: Monitor progress in defect resolution.
9. Regression Testing: Ensure changes don't introduce new defects.
10. Test Closure: Evaluate results and provide comprehensive summary reports.

Advanced Manual Testing Interview Questions for Experienced

40. What do you mean by defect detection percentage in software testing?

Defect detection percentage (DDP) is a type of testing metric. It indicates the effectiveness of a testing process by measuring the ratio of defects discovered before the release and reported after the release by customers.

For example, let's say, the QA has detected 70 defects during the testing cycle and the customer reported 20 more after the release. Then, DDP would be: 70/(70 + 20) = 72.1%

41. What does defect removal efficiency mean in software testing?

Defect removal efficiency (DRE) is an important testing metric. It is an indicator of the efficiency of the development team to fix issues before the release.

It gets measured as the ratio of defects fixed to total the number of issues discovered.

For example, let's say, there were 75 defects discovered during the test cycle while 62 of them got fixed by the development team at the time of measurement. The DRE would be 62/75 = 82.6%

Go through the Manual Testing Training to get a clear understanding of Weak AI and Strong AI.

42. As per your understanding, list down the key challenges of software testing.

Following are some of the key challenges of software testing:

- The lack of availability of standard documents to understand the application
- Lack of skilled testers
- Understanding the requirements: Testers require good listening and understanding capabilities to be able to communicate with the customers the application requirements.
- The decision-making ability to analyze when to stop testing
- Ability to work under time constraints
- Ability to decide which tests to execute first

- Testing the entire application using an optimized number of test cases

43. What is the average age of a defect in software testing?

Defect age is the time elapsed between the day the tester discovered a defect and the day the developer got it fixed.

While estimating the age of a defect, consider the following points:

- The day of birth of a defect is the day it got assigned and accepted by the development team.
- The issues which got dropped are out of the scope.
- Age can be both in hours or days.
- The end time is the day the defect got verified and closed, not just the day it got fixed by the development team.

Manual Testing Interview Questions For 2 Years Experience

44. What is a silk test and why should you use it?

Here are some facts about the silk test tool:

A specialized tool has been created for conducting regression and functional testing of an application.
It is used when we are testing Windows-based, Java, web, and traditional client/server applications.
Silk test helps in preparing the test plan and managing it to provide direct accessing of the database and validation of the field.

45. On the basis of which factors would you consider choosing automated testing over manual testing?

Choosing automation testing over manual testing depends on the following factors:

1. Tests require periodic execution.
2. Tests include repetitive steps.
3. Tests execute in a standard runtime environment.
4. Automation is expected to take less time.
5. Automation is increasing reusability.
6. Automation reports are available for every execution.
7. Small releases like service packs include a minor bug fix. In such cases, executing the regression test is sufficient for validation.

In such cases, executing the regression test is sufficient for validation.

46. How do you prioritize test cases when you have limited time for testing?

When it comes to prioritizing tests, I employ techniques such as risk-based testing and impact analysis. This entails giving priority to test cases based on their criticality to the applications functionality, frequency of usage, and potential consequences of failure. By addressing high-risk areas, I ensure that essential functionalities receive testing within the allocated time frame. This approach optimizes my testing efforts. Enhances the quality of the software product.

47. What are the key components of a test plan document, and why is it essential to the testing process?

A test plan document comprises the following elements:

1. Overview: This part gives a summary of the project's goals and the testing scope.
2. Approach: It describes the strategy that will be used for conducting tests.
3. Test Scope: In this part, we'll give you an overview of the project, its goals and the testing scope.
4. Test Schedule: The test schedule provides timelines and milestones for testing activities.
5. Resource Planning: This part focuses on allocating resources, tools, and environments required for testing purposes.
6. Test Deliverables:Test deliverables include all the documents and artifacts produced during testing.
7. Risk Management: Risk management involves strategies for identifying, assessing, and mitigating project risks.

A test plan is essential, as it serves as a guide for carrying out testing activities. It helps define roles and responsibilities, set expectations, and ensure alignment with project goals.

48. Can you explain the difference between smoke testing and sanity testing?

Smoke Testing:

Smoke testing is a form of software evaluation that aims to validate whether critical features of an application are working properly. It is usually carried out in the development phase immediately after deploying a build to ensure that vital functions are operational and that the application is sufficiently stable for testing.

The primary focus of smoke testing lies in identifying any issues that could impede the testing process.

Sanity Testing:

Sanity testing, a type of regression testing, focuses on assessing the functions or sections of the software after changes have been implemented. Its goal is to confirm that recent updates or repairs have not adversely affected the features of the software.

Compared to smoke testing, sanity testing has scope. Specifically targets areas affected by recent changes.

49. What are some common types of software defects you have encountered, and how do you typically categorize them?

Common types of software defects are:

1. Functional Defects: Failures to meet specified requirements.
2. Interface Defects: Issues with user interface elements.
3. Performance Defects: Problems related to software speed or resource usage.
4. Compatibility Defects: Incompatibility with certain systems or browsers.
5. Security Defects: Vulnerabilities compromising system security.
6. Usability Defects: Challenges users face while using the software.
7. Documentation Defects: Errors or omissions in documentation.

We can categorize defects by severity, priority, and impact, which aids in effective resolution and resource allocation.

50. How do you ensure adequate test coverage in your testing process?

Several strategies used to ensure adequate test coverage are:

1. Requirement Coverage: Ensure test cases cover all specified requirements and functionalities.
2. Risk Prioritization: Focus testing on high-risk areas or critical functionalities.
3. Boundary and Edge Cases: Test inputs at boundaries and edge conditions for accurate behavior.
4. Code Coverage Analysis: Measure the code exercise by the test suite to identify gaps.
5. Regular Reviews and Feedback: Review and enhance test cases periodically for improved coverage.

Manual Testing Interview Questions For 3 Years Experience

51. What are the key elements to consider while writing a bug report?

An ideal bug report should consist of the following key points:

- A unique ID
- Defect description: A short description of the bug
- Steps to reproduce: They include the detailed test steps to emulate the issue. They also provide the test data and the time when the error has occurred
- Environment: Add any system settings that could help in reproducing the issue
- Module/section of the application in which the error has occurred
- Severity
- Screenshots

- Responsible QA: This person is a point of contact in case you want to follow-up regarding this issue

52. Is there any difference between bug leakage and bug release?

Bug leakage: Bug leakage is when the bug is discovered by the end user/customer and missed by the testing team. It is a defect that exists in the application and not detected by the tester, which is eventually found by the customer/end user.

Bug release: A bug release is when a particular version of the software is released with a set of known bug(s). These bugs are usually of low severity/priority. It is done when a software company can afford the existence of bugs in the released software but not the time/cost for fixing it in that particular version.

53. What is exploratory testing?

Exploratory testing is an approach to software testing, where in testers learn simultaneously about the test design and test execution. In other words, it is a hands-on approach where testers are involved more in the test execution part than in planning.

Also, check out the blog on Test Data Management.

54. What is meant by system testing?

System testing is a black-box testing technique, used on a complete integrated system. It tests system compliance as per the requirement.

55. What are the benefits of test reports?

Test reports will help us find the current status of a project and its quality. This can help stakeholders and customers take necessary actions. The complete documentation of test reports will help analyze different phases of the project.

56. What is meant by latent defect?

A latent defect is a hidden defect in an application or software which cannot be identified by a user. However, this will not cause any failure to the application because the conditions will never be met.

57. Describe a scenario where you had to perform exploratory testing. What approach did you take, and what were the outcomes?

1. Recently, you had the opportunity to work on a project involving the testing of an e-commerce platform. To start off, you familiarize yourself with the features and functionalities of the application through testing. By sticking to predefined test cases you opted for a flexible approach and freely explored the application to identify any potential issues.
2. During your testing, you focused on aspects such as user registration, product search, and the checkout process. You simulated user scenarios and tried out various combinations of inputs and actions to uncover any unexpected behavior.
3. The results of your testing were quite significant. You came across usability issues, including navigation paths and inconsistencies, in error messaging. By reporting these issues, you were able to address them before launching the platform, thus improving the user experience.

58. How do you ensure that your test cases are robust and cover various scenarios adequately?

1. To ensure that the test cases are robust, you follow an approach. First, you thoroughly analyze the project requirements. Work closely with stakeholders. You make sure to understand the requirements in detail and identify user scenarios that match them with test cases.
2. Your goal is to cover a range of scenarios, including negative and boundary cases. This way, you can validate how the application behaves under certain conditions. You also consider edge cases and real-world user interactions to improve the coverage of your testing.
3. Regular reviews and feedback sessions with colleagues and experts in the domain help ensure that your test cases are comprehensive and cover all scenarios. Additionally, you use techniques, like equivalence partitioning and boundary value analysis, to refine and optimize your test cases.

59. What strategies do you use to manage and prioritize defects during the testing process?

1. During the testing process, you follow an approach to effectively handle and prioritize issues. First, you promptly document all identified problems by providing descriptions, step-by-step instructions for reproduction, and assessments of their severity.
2. You classify the issues according to how they affect the functionality of the application and assign them priorities such as critical, high, medium, or low. Critical defects that significantly impact core functionalities are given priority. Next in line are high-risk issues and those that have an impact on the user experience.
3. Through collaboration with developers and stakeholders, you ensure communication regarding the status of each issue, estimated timelines for resolution, and any dependencies involved. Regular triage meetings help determine which issues should be tackled first based on project

goals, risk factors involved, and available resources. This approach ensures that critical problems are addressed promptly in order to keep the project moving.

60. Can you discuss a challenging defect you encountered in your previous projects and how you resolved it?

1. In a project, I came across a problem with the payment processing module of a banking application. Users complained about failures when transferring funds between accounts, which caused discrepancies in transactions and left customers unhappy.
2. To tackle this issue, I conducted investigations and tests by examining server logs, transaction records and system interactions. By working with developers and conducting regression testing, I discovered a problem related to concurrency in the payment processing logic that only occurred under specific load conditions.
3. Fixing the defect required refactoring the code. Enhancing synchronization to ensure both thread safety and transaction integrity. Once the fixes were implemented, I carried out testing and validation to ensure that the solution was stable and correct. In the end, we successfully resolved the defect, restoring reliability and trustworthiness to the application.

61. How do you handle test data management and ensure data integrity during testing?

1. Effective management of test data is vital to ensure the accuracy and reliability of testing results. To handle test data efficiently, you employ a combination of strategies and tools, for generating, manipulating and cleaning up data.

2. You collaborate closely with stakeholders to identify scenarios for test data that encompass application functionalities and edge cases. Utilizing tools or scripts for data generation, you create datasets that represent user profiles, input scenarios and system configurations.
3. To maintain the integrity of the data, you ensure isolation and separation between test environments to prevent any cross contamination or interference. Additionally, you implement techniques, such as data masking or anonymization to safeguard information and comply with privacy regulations.
4. Regular sanity checks and validation procedures are conducted to verify the consistency and accuracy of the data throughout the testing lifecycle. Automated scripts or utilities for cleaning up data streamline management tasks while ensuring that test environments remain clean.
5. By adopting these practices, you guarantee that your test data remains reliable and relevant. This enables testing and accurate validation of application functionalities.

Manual Testing Interview Questions For 4 Years Experience

62. How do you perform automated testing in your environment?

Automation testing is a process of executing tests automatically. It reduces human intervention to a great extent. We use different test automation tools like QTP, Selenium, and WinRunner. Testing tools help in speeding up the testing tasks. These tools allow you to create test scripts to verify the application automatically and also to generate the test reports.

Preparing for a Job Interview! Check out our blog on Selenium Interview Questions now.

63. Is there any difference between quality assurance, quality control, and software testing. If so, what is it?

Quality Assurance (QA) refers to the planned and systematic way of monitoring the quality of the process which is followed to produce a quality product. QA tracks the test reports and modifies the process to meet the expectation.

Quality Control (QC) is relevant to the quality of the product. QC not only finds the defects but suggests improvements too. Thus, a process that is set by QA is implemented by QC. QC is the responsibility of the testing team.

Software testing is the process of ensuring that the product which is developed by developers meets the users' requirements. The aim of performing testing is to find bugs and make sure that they get fixed. Thus, it helps to maintain the quality of the product to be delivered to the customer.

64. Can you explain the principles of black-box testing and white-box testing, and when would you use each approach?

Black box testing evaluates software functionality without knowledge of the code structure. Testers verify system behavior by examining inputs and outputs simulating user interactions. This approach is ideal for validating requirements. Ensure that the software satisfies the user's expectations.

In contrast, white-box testing involves analyzing the structure and logic of the software. Testers have access to the source code and design, enabling them to create test cases based on code paths and coverage. This method is valuable for identifying errors in code implementation and ensuring coverage.

65. How do you verify data integrity and accuracy during manual testing of database-driven applications?

To ensure the integrity and accuracy of data, in testing applications driven by databases, you conduct checks:

1. Compare the data entered through the application, with the data stored in the database to make sure they are consistent.
2. Validate that data transformations and calculations are correct.
3. Verify that any updates or deletions of data are accurately reflected in the database.
4. Perform testing to ensure that data inputs within specified ranges are handled appropriately.

66. Can you explain the importance of SQL queries in manual testing, and provide an example of how you've used SQL queries to validate data in a testing scenario?

SQL queries play a crucial role in testing because they allow direct access to the database, making it easier to validate the accuracy and integrity of data. For example, in an e-commerce setting, you can utilize SQL queries to fetch order details from the database and compare them against expected values.

To illustrate, consider this query:

“SELECT * FROM Orders WHERE OrderID = ‘834’;”

This particular query retrieves order details so that you can ensure that the recorded information aligns with what’s anticipated based on the test scenario. By conducting validation, we guarantee that orders are processed and recorded correctly within the system.

67. Describe your experience with database testing. What types of SQL queries do you commonly use to retrieve and manipulate data for testing purposes?

In the field of database testing, you possess expertise in validating the integrity, functionality, and performance of data. You employ SQL queries to retrieve and manipulate data, guaranteeing its precision and reliability. This involves executing a range of query types, like retrieval, modification, aggregation, and join queries, to cover testing scenarios. Additionally, you verify the integrity constraints on the data. Conduct validation of applications driven by databases to ensure they fulfill all requirements.

Commonly used SQL queries in manual testing are:

1. SELECT: This query is used to retrieve data, for validation and verification purposes.
2. INSERT: It allows you to add test data to the database.
3. UPDATE: This query helps in modifying existing data and testing scenarios that involve updating data.
4. DELETE: It is used to remove test data once the testing process is complete.
5. JOIN: This query enables you to retrieve data from tables, which proves useful for testing.
6. AGGREGATE FUNCTIONS (e.g., COUNT, SUM, AVG): These functions are employed for performing calculations and validating data.
7. CONSTRAINTS (UNIQUE, NOT NULL): They ensure the integrity of the data. Enforce business rules.

68. Can you explain the difference between positive testing and negative testing in manual testing, and provide examples of when you would use each approach?

Testing can be categorized into two types:

- Positive testing
- Negative testing

Positive testing involves validating the system using inputs to ensure it performs as expected. An example of testing is entering an email address during the login process.

On the other hand, negative testing focuses on validating the system using inputs or unexpected conditions to assess how it handles errors gracefully. For instance, entering a password during a login falls under testing.

Positive testing predominantly aims to confirm expected behaviors, while negative testing plays a role in assessing error handling and resilience capabilities. Both types of testing are essential for achieving coverage in software testing.

Manual Testing Interview Questions For 5 Years Experience

69. Tell me about some of the essential qualities an experienced QA or Test Lead must possess.

A QA or Test Lead should have the following qualities:

1. Well-versed in software testing processes
2. Ability to accelerate teamwork to increase productivity
3. Improve coordination between QA and Dev engineers
4. Provide ideas to refine QA processes
5. Skill to conduct RCA meetings and draw conclusions
6. Excellent written and interpersonal communication skills
7. Ability to learn fast and to groom the team members

70. What is the difference between performance testing and monkey testing?

Performance Testing checks the speed, scalability, maybe even the stability characteristics of a system. Performance is identified with achieving response time, throughput, and resource-utilization levels that meet the performance objectives for a project or a product.

Monkey testing is a technique in software testing where the user tests the application by providing random inputs, checking the behavior of the application (or trying to crash the application).

71. How do you ensure effective communication and collaboration between testing and development teams during the software development lifecycle?

To ensure communication and collaboration, between the testing and development teams throughout the software development lifecycle, strategies include:

1. Meetings: We schedule meetings, such as daily stand-ups and sprint planning sessions, to encourage open communication and alignment regarding project goals and progress.
2. Documentation: We maintain regularly updated documentation, including test plans, user stories, and bug reports. This ensures that everyone involved has a shared understanding of the requirements and priorities.
3. Utilization of Collaboration Tools: We leverage collaboration tools like Slack, Microsoft Teams, or project management platforms such as Jira. These tools facilitate real time communication, file sharing, and issue tracking for collaboration.
4. Cross-Functional Teams: We foster an environment by promoting functional teams where testers and developers work closely together throughout the entire development process. This approach encourages a sense of ownership and collective responsibility for maintaining product quality.

5. Continuous Feedback: Timely feedback is crucial in our process. We provide feedback on test results, bug fixes and feature implementations to nurture a culture of improvement and iteration, within our teams. This enables us to address issues while enhancing product quality.

72. What strategies do you employ for ensuring comprehensive test coverage, particularly in large and complex software systems?

To ensure testing of complex software systems it is important to follow a systematic approach. Here are some strategies that can be employed:

1. Requirement Analysis: Carefully examine the project requirements to identify all aspects, both non functional that require testing.
2. Risk Based Testing: Prioritize testing efforts based on risk assessment. This involves focusing on functionalities and areas that're more likely to have defects.
3. Test Planning: Develop test plans that outline the objectives, scope, available resources and timelines. It is essential to cover all aspects of the testing process.

4. Test Case Design: Create test cases that encompass a range of scenarios such as positive, negative, boundary and edge cases.
5. Automation Testing: Implement automated tests for tasks like regression testing and performance testing. This does not increase test coverage. Also improves efficiency.

73. How do you ensure thorough test coverage in a manual testing scenario, particularly when dealing with complex software functionalities?

To achieve test coverage in a testing scenario particularly when dealing with complex software functionalities there are several strategies that can be employed:

1. Understanding the Requirements; It is crucial to comprehend the project requirements in order to identify functionalities and potential edge cases that necessitate testing.
2. Creating a Comprehensive Test Plan; Developing a test plan is essential. This plan should outline objectives, scope, available resources and timelines. Test scenarios should be prioritized based on their criticality and complexity.
3. Designing Detailed Test Cases; The creation of test cases is paramount. These test cases should encompass scenarios such, as negative, boundary and edge cases.
4. Conducting Exploratory Testing: This approach helps us discover any defects that might have slipped through unnoticed otherwise.
5. Implementing Risk Based Testing; Prioritizing testing efforts based on risk assessment is essential for resource allocation. Critical functionalities and areas prone to defects should receive attention in terms of time and resources in order to ensure coverage.

74. How do you use SQL queries to validate data integrity and accuracy during manual testing, and can

you provide an example of how you've used SQL queries to verify data in a testing scenario?

To validate data integrity and accuracy during manual testing using SQL queries, testers follow a systematic approach:

1. Understanding the Data Requirements: It is important to have an understanding of what data needed and what outcomes are expected for the testing scenario.
2. Creating SQL Queries: Develop SQL queries that can retrieve the data from the database tables based on the requirements of the testing scenario.
3. Comparing Data: Execute the SQL queries to retrieve data from the database and then compare it with the expected data in order to ensure accuracy and integrity.
4. Verifying Constraints: Validate data integrity constraints, such as ensuring uniqueness, maintaining relationships and confirming appropriate data types. This can be done using SQL queries.
5. Example Scenario: Let's take an example related to an e-commerce application. Using SQL queries, we can verify that when a successful purchase transaction occurs there is a decrease in the quantity of items in the inventory table.

To do this, we will write a SQL query to get the quantity of a product from the database. After simulating a purchase transaction, we would execute another query to confirm that the quantity has been decremented by an amount.

75. When performing SQL manual testing, what are some common challenges you've faced, and how have you overcome them to ensure effective testing of database-driven applications?

When it comes to testing of SQL, you encounter a common challenges:

1. Dealing with Complex Query Execution: Writing and executing SQL queries to validate data scenarios can be quite a challenge.
2. Ensuring Data Integrity: It can be tricky to maintain accurate test data throughout the testing process, especially when dealing with datasets.
3. Optimizing Performance: Identifying and optimizing SQL queries for performance can pose a challenge in situations involving large databases or complex data retrieval operations.
4. Managing Data Dependencies: Handling data dependencies between test cases or scenarios can be tricky, particularly when making changes to interconnected data.

To tackle these challenges and ensure the testing of applications driven by databases, you utilize a range of strategies, including:

1. Thorough Planning: Meticulously plan test scenarios and queries in advance to address obstacles and achieve coverage.
2. Test Data Management: Maintain organized and representative sets of test data, utilizing appropriate tools or scripts, for data generation and manipulation as required.
3. Improving Query Performance: Optimize SQL queries for performance by analyzing query execution plans, employing indexing strategies, and optimizing database configurations.
4. Version Control Implementation: Implement version control for database schemas and scripts to track changes and enable rollback if necessary during the testing phase.
5. Collaboration: Close collaboration with developers and stakeholders is crucial. By understanding the application logic, data models, and requirements, we ensure alignment and effective communication, throughout the testing process.

Manual Testing Scenario-Based Interview Questions

76. How would you test the login page to ensure it properly handles invalid credentials? Describe the steps you would take and any expected outcomes.

To ensure the login page handles invalid credentials effectively:

1. Set up the Test Environment: Ensure the application is deployed and accessible for testing.
2. Define Test Cases: Create test cases covering various scenarios of invalid credentials, such as incorrect username/password and blank fields.
3. Execute Tests: Enter invalid credentials into the login form and submit it to observe the application's response.
4. Observe Behavior: Verify if appropriate error messages are displayed for different types of invalid credentials.
5. Anticipated Results: Be prepared to receive error notifications that specify the type of input, like "Username or password's invalid" or "Password entered is incorrect."
6. Document Results: Record observed behavior and any deviations from expected outcomes.
7. Repeat Testing: Perform testing for all identified scenarios to ensure thorough coverage.

77. How would you test the product search functionality to ensure it returns accurate results? Explain your approach and any test cases you would consider.

To ensure the product search functionality delivers accurate results, employ the following SEO-friendly approach:

- Test Preparation: Ensure accessibility of the application and the availability of the search feature.
- Positive Test Scenarios:

1. Valid Search Term: Test with precise search terms matching existing products for accurate results.
2. Partial Match: Validate the functionality with partial search terms, ensuring relevant products are displayed.
3. Case Insensitivity: Confirm case insensitivity in search terms for a seamless user experience.
4. Category-based Search: Validate search within specific categories for tailored results.
5. Price Range Search: Ensure the accurate display of products falling within specified price ranges.

- Negative Test Scenarios:
    1. Invalid Search Term: Test with nonexistent or invalid search terms, ensuring appropriate error messages are shown.
    2. Empty Search: Validate the application's response when the search field is empty, prompting users to input search terms.
    3. Special Characters Handling: Verify the graceful handling of special characters, providing meaningful feedback.
- Pagination Testing:
    1. Efficient Navigation: Assess pagination functionality with large result sets, ensuring smooth navigation between pages.
- Performance Evaluation:
    1. Load Testing: Gauge performance under varying loads to ensure swift response times.
    2. Response Time Analysis: Measure response times, adhering to acceptable thresholds for an optimal user experience.
- Cross Device Testing:
    1. Consistent Experience: Verify functionality across browsers and devices to maintain consistency and accessibility.

78. Walk me through the testing process for the checkout process. What aspects would you focus on to ensure a smooth and error-free transaction?

When aiming for an accurate checkout experience, keep these points in mind:

- Preparation: Ensure the website is accessible and the checkout feature is operational.
- Test Scenarios:
    1. Adding Items to Cart: Confirm the ability to add items from product pages.
    2. Cart Review: Validate the accuracy of displayed items, quantities, and prices.
    3. User Authentication: Test login and guest checkout options.
    4. Address Entry: Verify address form validation and error handling.
    5. Shipping Options: Ensure correct options based on the entered address.
    6. Payment Methods: Test secure processing and error handling for various payment methods.
    7. Order Summary: Confirm the accuracy of the order details.
    8. Order Confirmation: Validate the generation of order pages and emails.
- Edge Cases:
    1. Out-of-stock Items: Test handling for unavailable items during checkout.
    2. Partial Payments: Validate split payment capabilities.
    3. Session Timeouts: Ensure data retention during session timeouts.
    4. Network Failures: Test error recovery during transaction processing.
- Performance Testing:
    1. Load Testing: Assess responsiveness under different loads.
    2. Transaction Time Analysis: Measure processing times to ensure efficiency.
- Security Testing:
    1. Data Encryption: Verify the secure transmission of sensitive information.

2. Payment Gateway Integration: Validate integration with secure payment gateways.

- Cross Device Testing:
    1. Browser Compatibility: Test across browsers for consistency.
    2. Device Compatibility: Validate usability on different devices for responsiveness.

79. How would you test the user registration process to ensure it functions correctly? What validations and verifications would you perform?

To ensure the user registration process works flawlessly, consider the following steps:

- Setup: Ensure the registration feature is accessible and operational.
- Positive Tests:
    1. Valid Registration: Verify the successful registration with the correct details.
    2. Username Uniqueness: Confirm usernames are unique.
    3. Password Strength: Validate password complexity requirements for security.
    4. Email Validation: Ensure correct email formatting and validation.
    5. Confirmation Email: Verify users receive confirmation emails post-registration.
- Negative Tests:
    1. Invalid Email Format: Test registration with incorrectly formatted emails.
    2. Existing Username: Attempt registration with an already used username.
    3. Weak Password: Test with passwords that do not meet complexity criteria.

4. Empty Fields: Validate mandatory fields and corresponding error messages for empty inputs.

- Edge Cases:
    1. Long Username/Password: Test with unusually long inputs for system stability.
    2. Special Characters: Validate registration with special characters in inputs.
    3. Session Timeout: Ensure registration can be completed after session timeouts.
- Performance:
    1. Load Testing: Assess registration performance under different loads.
    2. Response Time: Measure response times for prompt feedback.
- Security:
    1. Data Encryption: Ensure sensitive data encryption during storage.
    2. SQL Injection: Test for protection against SQL injection attacks.
- Cross Device Testing:
    1. Browser Compatibility: Validate across browsers for consistency.
    2. Device Compatibility: Test on various devices for responsiveness.

80. Describe your approach to testing form submissions to ensure data validation. How would you handle scenarios where invalid data is entered?

To effectively test form submissions for data validation, consider the following SEO-friendly approach:

- Understanding Requirements: Gain clarity on form requirements, including mandatory fields and validation rules.

- Positive Test Cases:
    1. Enter valid data according to the specified requirements.
    2. Confirm successful form submission and accurate data processing.
- Negative Test Cases:
    1. Input invalid data that violates validation rules.
    2. Validate appropriate error messages indicating validation failures.
- Edge Cases:
    1. Test extreme or boundary scenarios to ensure robust validation.
    2. Assess behavior with unexpected inputs or special characters.
- Cross device Testing:
    1. Validate form submissions across browsers and devices.
    2. Ensure responsiveness and usability across several screen sizes.
- Handling Invalid Data Scenarios:
    1. Provide clear error messages with corrective actions.
    2. Implement client-side validation for immediate feedback.
    3. Validate data on the client and server sides for integrity.
    4. Log and track validation errors for resolution.

81. How do you test error handling mechanisms to ensure users receive meaningful error messages? Provide examples of error scenarios you would test.

To ensure users receive meaningful error messages, follow this SEO-friendly approach to test error handling mechanisms:

- Understand Requirements: Clarify expected behaviors and error messages outlined in requirements or design documents.
- Positive Test Cases:

1. Confirm that no unnecessary error messages are displayed during valid actions like form submissions or navigation.

- Negative Test Cases:
    1. Intentionally trigger errors by providing incorrect inputs or accessing restricted features.
    2. Validate appropriate error messages explaining errors and suggesting fixes.
- Edge Cases:
    1. Test extreme scenarios like data exceeding limits or unexpected server responses.
    2. Evaluate application responses and error messages in these situations.
- Cross Device Testing:
    1. Ensure error handling consistency across browsers and devices.
    2. Verify usability on different screen sizes.
- Specific Error Scenarios:
    1. Test scenarios like invalid login credentials or unauthorized resource access.
    2. Validate the handling of network issues or input validation errors in forms.

82. Explain your approach to compatibility testing. How would you ensure the website functions correctly on various browsers and devices?

To ensure your website functions seamlessly across various browsers and devices, follow this SEO-friendly approach to compatibility testing:

1. Creating a chart that lists the browsers and devices supported is tailored to your audience and industry norms.
2. Using tools like BrowserStack, CrossBrowserTesting, or Sauce Labs for browser testing to mimic how your website behaves in different settings.

3. Manually testing on browsers and devices to confirm appeal, functionality, and performance.
4. Running tests on each browser/device combination to ensure that all crucial website features function correctly.
5. Checking design to guarantee adaptation to different screen sizes, resolutions, and orientations.
6. Testing compatibility with browser plugins/extensions to prevent any disruptions in website functionality.
7. Evaluating performance metrics such as load time and responsiveness on platforms to pinpoint and resolve performance issues.
8. Validating CSS and JavaScript compatibility across browsers for appearance and functionality.
9. Conducting regression testing to confirm the resolution of identified problems and prevent their recurrence on browsers/devices.
10. Verifying accessibility features like screen reader compatibility and keyboard navigation for adherence to accessibility standards.

Salary in Manual Testing based on Skills

Job Role	Average Salary in India	Average Salary in the USA
Manual Tester – Experience (0 – 9) years	Minimum – 6.5 LPA	Minimum – 66,300 USD
	Average – 12 LPA	Average – 82,500 USD
	Highest – 22 LPA	Highest – 107,250 USD

Manual Testing Job Trends in 2024

According to the Bureau of Labor Statistics US, the employment of manual testers will be projected to grow by 25% from 2022 to 2032.

- Global Demand: With more than 27,000 open jobs on LinkedIn in the United States and more than 9,000 open jobs on LinkedIn in India.
- Growth Projections: The growth suggested by the Bureau of Labor Statistics is 25% in the field of Manual testing, which might surpass all other occupation fields' growth by 8%.

Job Opportunities in Manual Testing

Job Role	Description
Quality Assurance Analyst	Designing test plans and strategies Creating and executing test cases Identifying, documenting, and tracking software defects Analyzing test results and providing feedback to stakeholders Ensuring adherence to quality standards

Software Tester	Reviewing software requirements and specifications Writing and executing test cases Verifying software functionality Reporting and documenting defects Collaborating with development teams to resolve issues
Test Engineer	Developing test plans and test cases Executing manual tests Performing regression testing Conducting exploratory testing Providing input on test automation Participating in test strategy and planning meetings

QA Specialist	Evaluating software products and systems Identifying areas for improvement in testing processes Developing and implementing QA standards and procedures Mentoring junior testers Participating in continuous improvement initiatives

Roles and Responsibilities in Manual Testing

According to the job posted on Naukri.com by GSR Business Services

Role: Manual Tester

1. Responsibilities
    - Work together with internal Siemens teams to develop, verify, and maintain software.
    - Take part in software verification and validation as a member of a functional team.
    - Contribute to estimating, planning, and verifying/validating processes using methods.
    - Ensure the integration of software components while maintaining quality standards and meeting deadlines.
    - Record testing outcomes and actively support improvements.
2. Skill Required:
    - Knowing test management tools such as IBM Jazz, JIRA and Quality Center.

- Experienced in scripting languages like Javascript, Python, Perl or TCL.
- Hands-on experience with tools like Robot or Selenium.
- Worked on cross platform product verification.
- Proficient in using Git.

Conclusion

I hope this set of Manual Testing Interview Questions will help you prepare for your interviews. Best of luck!

If you are looking to embark on a manual testing journey that will uplift your career, check out Intellipaat's Manual testing course, or enroll in Intellipaat's Executive Post Graduate certification in AI and ML for an enriching learning experience and career growth.

Got any questions regarding manual testing? Post your query in the Intellipaat Community space, and we will get back to you.

