Q A TESTING (MANUAL AND AUTOMATION) QUESTIONS

---

1. Explain what is software testing.

It is the process of analyzing any given piece of software to determine if it meets shareholders' needs as well as detecting defects, and ascertaining the item's overall quality by measuring its performance, features, quality, utility, and completeness. Bottom line, it's quality control.

2. What is quality control, and how does it differ from quality assurance?

Quality control is the process 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.

3. What exactly is manual software testing, and how does it differ from automated software testing?

Manual software testing is a process where human testers manually run test cases, then generate the resulting test reports. With automation software testing, these functions are executed by automation tools such as test scripts and code. The tester takes the end user's role to determine how well the app works.

4. What are the advantages of manual testing?

Manual testing's strengths are:
It's cheaper
You get visual feedback that's accurate and quick
It's ideal for testing minor changes
It's perfect for ad hoc testing
Testers don't have to know anything about automation tools
It's great for testing UI's

5. On the other hand, what are the drawbacks to manual testing?

Manual testing's weaknesses are:
Susceptible to human error
Some tasks may be difficult to accomplish manually, requiring more time to complete
The cost adds up, so it's more expensive in the long run
You cannot record the manual testing process, so it's hard to replicate it

6. What kind of skills are needed for someone to become a software tester?

Software testers need skills such as:
Problem-solving skills
Excellent written and verbal communication skills
Detail-oriented
Able to handle the pressure
Can work solo or as a team member equally well

Organizational skills
Related technical skills

7. Explain what is SDLC.

This is an acronym for Software Development Life Cycle and encompasses all of the stages of software development, including requirement gathering and analysis, designing, coding, testing, deployment, and maintenance.

8. What is a test case?

Test case is used to check whether an application complies with its requirements. It is a documented set of circumstances including prerequisites, input values, and expected outcomes.

9. What is a test scenario?

A test scenario is derived from a use case. It's used to test an application's feature from beginning to end. Multiple test cases can be accommodated by a single test scenario. When there is a time constraint during testing, scenario testing comes in handy.

10. What is a test plan?

A test plan is a formal document that specifies the scope of testing, the method to be used, the resources needed, and the estimated time to complete the testing process. It is derived from the specifications (Software Requirement Specifications).

11. What is test data?

Test data is information that is used to test software with various inputs and determine whether the resulting output matches the intended result. This data is generated based on the needs of the company.

12. What is a test script?

An automated test case created in any programming or scripting language is known as a test script. These are essentially a collection of instructions for evaluating an application's functionality.

13. What types of manual testing are there? Break them down.

Manual testing is broken down into:
Black Box
White Box
Integration
Unit
System
Acceptance

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

Software testers employ black-box testing when they do not know the internal architecture or code structure. The techniques are:
Equivalence Partitioning

Boundary value analysis
Cause-effect graphing

15. What is white box testing and its various techniques?
Unlike black-box testing, white box involves analyzing the system's internal architecture and/or its implementation, in addition to its source code quality. It's techniques are:
Statement Coverage
Decision Coverage
So far, if you have any doubts about these Manual testing interview questions/ software testing interview questions, please ask in the comment section below.
Also Read: Top 16 Types of Software Testing and How to Use Them

16. Explain the difference between alpha testing and beta testing.
Alpha testing is at the developer's site before release. Potential clients conduct beta testing at their websites.

17. What's the difference between verification and validation?
Verification evaluates the software at the development phase, ascertaining whether or not a product meets the expected requirements. On the other hand, validation evaluates the software after the development phase, making it sure it meets the requirements of the customer.

18. What's a testbed?
It's not furniture. A testbed is an environment used for testing an application, including the hardware as well as any software needed to run the program to be tested.

19. What is Sanity testing?
Sanity testing is testing done at the release level to test the main functionalities. It's also considered an aspect of regression testing.
Got a question for us? Please mention it in the comments section on this Manual Testing Interview Questions article and we will get back to you.
Also Read: Sanity Testing Vs Smoke Testing

20. When should developers implement configuration management procedures?
This should be done during test planning.

21. List the four different test levels
The four levels are:
Unit/component/program/module testing
Integration testing
System testing
Acceptance testing

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

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

23. What about the difference between an error and a failure?

If a program can't run or be compiled during development, it's an error. If an end-user discovers an issue with the software, it's a failure.

24. What's GUI testing?

This tests the interface between the software and the end-user. Short for Graphics User Interface.

25. When should testing end?

There are a few criteria for ending testing:
The bug rate has fallen below an agreed-upon level
The testing or release deadlines have arrived
The testing budget is out of funds
A certain percentage of test cases have passed
The alpha or beta testing periods have ended
Code, functionality, or requirements coverage have been met at a declared point
These were some basic manual testing interview questions. In the coming section, we bring to you some advanced level manual testing interview questions.

26. Why is Software Testing Required?

Software testing is required to ensure the quality and reliability of a software product.
Testing helps to uncover any bugs, errors, or other issues in the software so that they can be addressed and fixed before the product is released.
Testing also ensures that the software meets all the requirements specified by the customer and works as expected.
Finally, testing helps to ensure that the software is secure and can withstand malicious attacks.

27. What are the different levels of manual testing?

The different levels of manual testing are:
Unit Testing
Integration Testing
System Testing
User Acceptance Testing
Performance Testing
Security Testing
Compatibility Testing
Usability Testing
Installation Testing
Smoke testing
Sanity testing

Regression Testing

28. Explain the procedure for manual testing?

Manual testing is a process of identifying bugs and errors in a software product without the use of automated tools. The procedure for manual testing is as follows:

Identify the scope of testing: The first step of manual testing is to identify the scope of testing. The range could be a specific module, functionality, feature, or end-to-end system.

Design test cases: The next step is to design test cases based on the identified scope. The test cases should include test scenarios, data, expected results, and all other details necessary to perform the tests.

Execute the test cases: After designing the test cases, the testers execute them to find any discrepancies between the expected and actual results.

Record the results: While performing the tests, the testers should record the results for further analysis.

29. What is the test case?

A test case is a set of conditions or variables under which a tester will determine whether a software system or one of its features is working as it was originally established for it to do. It may take the form of input, action, or environmental conditions. In addition, a test case includes requirements, test steps, verification steps, prerequisites, outputs, and actual results.

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

Documentation is an integral part of manual testing. It is essential to document all steps taken in the testing process to ensure thorough test coverage and accurate results. Documentation provides an audit trail, which can be used to evaluate past test results and identify areas for improvement. Additionally, it is a reference for other testers who may be unfamiliar with the system or application under test.

31. What are the different types of Software testing?

Software testing is classified into two main categories.

Functional testing

Non-Functional testing

32. Explain Functional Testing

Functional testing is a type of black-box testing. It focuses on the software's functional requirements rather than its internal implementation. A functional requirement refers to the system's needed behaviour in terms of input and output.

It checks the software against the functional requirements or specification, ignoring non-functional characteristics like performance, usability, and dependability.

The purpose of functional testing is to ensure that the software up to snuff in terms of functionality and to solve the difficulties of its target users.

Some of the types of functional Testing are -

Unit Testing

Integration Testing

Regression Testing

System Testing
Smoke Testing
Performance Testing
Stress Testing

33. Explain Non functional testing

Non-functional testing examines the system's non-functional requirements, which are characteristics or qualities of the system that the client has specifically requested. Performance, security, scalability, and usability are among them.

Functional testing is followed by non-functional testing. It examines aspects that are unrelated to the software's functional requirements. Non-functional testing assures that the programme is safe, scalable, and fast, and that it will not crash under excessive pressure.

34. Mention a few advantages of Automated testing.

The following are some major advantages of automated testing -

- Automated test execution is quick and saves a significant amount of time.
- Human mistakes are eliminated during testing when test scripts are carefully prepared.
- CI tools like Jenkins, which may also be set to distribute daily test results to key stakeholders, can be used to schedule test execution for a nightly run.
- Automation testing uses a lot less resources. Test execution requires nearly no time from QAs once the tests have been automated. QA bandwidth can be used for other exploratory work.

35. What is Regression Testing?

Regression Testing is a full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine.

36. What is Test Harness?

A test harness is a collection of software and test data used to put a programme unit to the test by running it under various conditions such as stress, load, and data-driven data while monitoring its behaviour and outputs.

37. Differentiate between Positive and Negative Testing

Positive Testing

Positive testing ensures that your software performs as expected. The test fails if an error occurs during positive testing.

In this testing, the tester always looks for a single set of valid data.

Negative Testing

Negative testing guarantees that your app can gracefully deal with unexpected user behaviour or incorrect input.

Testers use as much ingenuity as possible when validating the app against erroneous data.

38. What is a Critical Bug?

A critical bug is one that has the potential to affect the bulk of an application's functioning. It indicates that a significant portion of functionality or a critical system component is utterly broken, with no way to proceed. The application cannot be delivered to end users until the critical bug has been fixed.

39. What is Test Closure?

Test Closure is a document that summarises all of the tests performed throughout the software development life cycle, as well as a full analysis of the defects fixed and errors discovered. The total number of experiments, the total number of experiments executed, the total number of flaws detected, the total number of defects settled, the total number of bugs not settled, the total number of bugs rejected, and so on are all included in this memo.

40. Explain the defect life cycle.

A defect life cycle is a process by which a defect progresses through numerous stages over the course of its existence. The cycle begins when a fault is discovered and concludes when the defect is closed after it has been verified that it will not be recreated.

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

According to the pesticide paradox, if the same tests are done repeatedly, the same test cases will eventually stop finding new bugs. Developers will be especially cautious in regions where testers discovered more flaws, and they may overlookPositive and Negative Testing?
other areas. Methods for avoiding the pesticide conundrum include:
To create a completely new set of test cases to put various aspects of the software to the test.
To create new test cases and incorporate them into existing test cases.
It is possible to detect more flaws in areas where defect levels have decreased using these methods.

42. What is API testing?

API testing is a sort of software testing that entails evaluating application programming interfaces (APIs) to see if they meet functionality, reliability, performance, and security requirements. Simply put, API testing is designed to detect defects, inconsistencies, or departures from an API's expected behaviour. Typically, applications are divided into three layers:
The user interface is also known as the presentation layer.
For business logical processing, the Business Layer or application user interface is used.
API testing is done at the most vital and important layer of software architecture, the Business Layer, for modelling and manipulating data.

43. What is System testing?

System testing is a type of testing in which the entire software is tested. System testing examines the application's compliance with its business requirements.

44. What is Acceptance testing?

Acceptance testing is a type of testing done by a possible end-user or customer to see if the software meets the business requirements and can be used.

45. Differentiate between bug leakage and bug release

Bug Leakage - When tested software is pushed into the market and the end-user discovers defects, this is known as bug leakage. These are bugs that the testing team overlooked throughout the testing phase.

Bug Release - When a certain version of software is launched into the market with some known bugs that are expected to be fixed in later versions, this is known as a bug release. These are low-priority issues that are highlighted in the release notes when sharing with end-users.

46. What do you mean by Defect Triage?

Defect triage is a procedure in which defects are prioritised depending on a variety of characteristics such as severity, risk, and the amount of time it will take to fix the fault. The defect triage meeting brings together several stakeholders - the development team, testing team, project manager, BAs, and so on – to determine the order in which defects should be fixed

47. What is Integration Testing? What are its types?

Integration testing is performed after unit testing. We test a group of linked modules in integration testing. Its goal is to identify faults with module interaction.

The following are the types of integration testing -

- Big Bang Integration Testing — After all of the modules have been merged, big bang integration testing begins.
- Top-down Integration Testing — In top-down integration, testing and integration begin at the top and work their way down.
- Bottom-up Integration Testing — In bottom-up integration testing, lower-level modules are tested before moving up the hierarchy to higher-level modules.
- Hybrid Integration Testing — Hybrid integration testing combines top-down and bottom-up integration testing techniques. The integration with this approach starts at the middle layer, and testing is done in both directions.

48. What is a stub?

Many times, when top-down integration testing is performed, lower-level modules are not produced until top-level modules are tested and integrated. Stubs or dummy modules are used in these circumstances to emulate module behaviour by delivering a hard-coded or predicted result based on the input variables.

49. What is code coverage?

The quantity of code covered by the test scripts is referred to as code coverage. It conveys the scope of the test suite's coverage of the application.

50. What is a cause-effect graph?

A cause-effect graph testing technique is a black-box test design technique that uses a graphical representation of the input (cause) and output (effect) to construct the test. This method employs a variety of notations to describe AND, OR, NOT, and other relationships between the input and output conditions.

51. Explain equivalence class partitioning.

Equivalence class partitioning is a black-box testing technique based on specifications. A set of input data that defines multiple test conditions is partitioned into logically comparable groups in equivalence class partitioning, so that utilising even a single test data from the group for testing can be considered as similar to using all the other data in that group.

52. What is boundary value analysis?

The border values of the classes of the equivalence class partitioning are used as input to the test cases in boundary value analysis, which is a software testing technique for designing test cases.

53. What is your approach towards a severely buggy program? How would you handle it?

In such cases, the best course of action is for testers to go through the process of reporting any flaws or blocking-type issues that arise, with an emphasis on critical bugs. Because this sort of crisis might result in serious issues such as insufficient unit or integration testing, poor design, wrong build or release methods, and so on, management should be contacted and given documentation as proof of the problem.

54. What if an organization's growth is so rapid that standard testing procedures are no longer feasible? What should you do in such a situation?

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

Hire people who are good at what they do.

Quality issues should be ‘fiercely prioritised' by management, with a constant focus on the client.Everyone in the company should understand what the term "quality" implies to the end-user.

55. When can you say for sure that the code has met its specifications?

Most businesses have coding "standards" that all developers are expected to follow, but everyone has their own opinion on what is best, as well as how many regulations are too many or too few. There are many methods available, such as a traceability matrix, to guarantee that requirements are linked to test cases. And when all of the test cases pass, that means the code satisfies the requirement.

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

Manual testing is the process of manually testing software for defects. It requires a tester to manually execute the test steps and compare the actual and expected results. Automation testing uses special software to control the execution of tests and compare the results with the

desired results. As a result, automation testing is much faster than manual testing and can reduce the time required to complete a test cycle.

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

Manual testing should be used over automation testing when the tests are particular or require human interpretation. Manual testing is also better suited for exploratory testing, usability testing, and testing on multiple operating systems or unique hardware.

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

The phases involved in the Software Testing Life Cycle are:

- Test Planning
- Test Analysis
- Test Design
- Test Implementation
- Test Execution
- Test Results Analysis
- Test Closure

59. What makes a good test engineer?

A good test engineer is detail-oriented and organized, has excellent problem-solving skills, and can produce high-quality work quickly and efficiently. And also should have strong communication and collaboration skills and be an outstanding team player. They also need to be up to date on the latest technologies and software trends and be able to apply them to their testing process.

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

System testing is a type of software testing that evaluates a complete and fully integrated software product. It verifies that the software meets the requirements specified in the design and the system-level technical specifications. System testing also identifies any weaknesses, errors, or bugs.

Integration testing is software testing that verifies the interactions between two or more system components. It is performed after unit testing and before system testing. It checks how components interact with each other and how they fit together. Integration testing is necessary to ensure that the components of the system work together as expected.

61. What is Defect Cascading in Software Testing?

Defect cascading is a type of software testing issue in which the result of a defect in one part of the system causes other defects or problems to occur in other parts of the system. This cascading causes a chain reaction of errors, making it difficult to trace the source of the problem. Defect cascading can lead to many issues, from minor performance slowdowns to significant system crashes, making it a severe risk to software developers and testers.

62. What does the term ‘quality' mean when testing?

Quality in testing refers to the degree to which a product meets its intended requirements, as well as the degree to which it satisfies customer needs and expectations. It includes both the

functional and non-functional aspects of the product. Quality assurance is ensuring that the product meets its requirements, while quality control focuses on testing to ensure that the product meets its needs.

63. What are the Experience-based testing techniques?

Experience-based testing techniques include:

- Exploratory Testing
- Error Guessing
- Adhoc Testing
- Checklist-based Testing
- Exploit-based Testing
- Session-based Testing
- Alpha Testing
- Beta Testing
- User Acceptance Testing
- Usability Testing.

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

A top-down and bottom-up approach in testing refers to the order of testing.

Top-down testing begins at the highest level and works downward. Thus, each higher-level component is tested in isolation from the lower-level components.

Bottom-up testing starts at the lowest level and works upward. Thus, each lower-level component is tested in isolation from higher-level components.

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

Smoke testing is a high-level test used to ensure the most critical functions of a software system are working correctly. It is a quick test that can be used to determine whether it is worth investing time and energy into further, more extensive testing.

Sanity testing is a more specific test used to check that recent changes to a system have not caused any new, unwanted behavior. It ensures that basic features are still functioning as expected after minor changes have been made.

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

Type of Testing	Description
Static Testing	Testing performed without executing the code. Involves reviews, inspections, and walkthroughs.
Dynamic Testing	Testing involving code execution to determine functionality. Includes unit testing, integration testing, and acceptance testing.

67. How will you determine when to stop testing?

When testing, it is vital to determine when to stop to prevent wasting resources. When deciding when to stop testing, then you should consider the following criteria:

Desired levels of quality

Adherence to timelines and budget

Number of defects found

Number of test cases that have been completed

Risk factors associated with the project

Once these criteria have been met, you can stop your testing.

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

When requirements are yet to freeze, the best approach is to use an agile development methodology, such as Scrum.

The first step would be to hold requirements gathering meetings with all stakeholders to understand the product's purpose and desired outcomes. The next step would be to break up the project into individual, manageable user stories.

From there, we would prioritize the user stories and assign them to sprint for development.

As the project progresses, we continually test the product using techniques such as unit tests, integration tests, user acceptance tests, and system testing. In addition, as requirements change, we will update our tests to ensure the product meets the desired outcomes.

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

The following steps are the considering cases:

When the test requires repetitive steps:

Automated testing is ideal for running tests requiring multiple iterations or repeating the same actions repeatedly.

When the test requires a large amount of data:

Automated testing can quickly insert large amounts of data into the tested system.

When the test requires multiple environments:

Automated testing can easily be configured to test systems in various domains, such as multiple operating systems, browsers, and devices.

When the test requires precise timing:

Automated tests can be programmed to run precisely, ensuring that each test step is performed at the exact time it needs to be.

When the test requires multiple users:

Automated testing can simulate multiple users accessing the system simultaneously, allowing for more realistic testing.

70. What is 'configuration management'?

Configuration management is managing, tracking, and controlling changes to a system's software, hardware, or network configuration. Configuration management can maintain the integrity of a system and ensure that it is secure, stable, and compliant with organizational policies. The primary goals of configuration management are to ensure system reliability, maintain system availability, and improve system performance.

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

No, system testing is typically carried out at the end of the development process, after integration and user acceptance testing.

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

Here are the top 10 best test case practices:
Develop test cases that are clear, concise, and to the point.
Ensure that the test cases challenge the software's functionality in all dimensions.
Make sure that the test cases cover all the requirements.
Develop repeatable test cases that can be automated when necessary.
Develop test cases that are independent of each other.
Use meaningful and descriptive names for test cases.
Record the results of test cases for future reference.
Make sure that the test cases are modular and can be reused.
Perform reviews of the test cases to ensure accuracy and completeness.
Document the test cases in a standard format.

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

Boundary value analysis provides suitable test cases because it ensures that the boundaries of input and output values are tested, making it easier to identify edge cases. Testing these edge cases ensures that your system is robust and can handle any unexpected input or output values.

74. Why is it impossible to test a program thoroughly or 100% bug-free?

It is impossible to test a program thoroughly or 100% bug-free because it is impossible to anticipate and test every possible combination of inputs, environments, and states a program might encounter.

75. Can automation testing replace manual testing?

No, automation testing cannot fully replace manual testing. Automation testing is designed to supplement manual testing, not replace it. Automation testing can automate repetitive, tedious test cases and make the testing process more efficient. However, it cannot replace manual testing completely, as some tests can only be performed manually.

76. What is Software Testing?

Software Testing is a process used to identify developed software’s correctness, completeness, and quality. It includes a series of activities conducted to find software errors so that it can be corrected before the product is released to the market.

77 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 to make 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.

78. 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.

79. 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.

80. 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.

81. 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

82. 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.

83. 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.

84. 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.

85. What is a test scenario?

A test scenario in software testing is a detailed description of a specific functionality to be tested. It outlines conditions, inputs, and expected outcomes for testers to follow. By replicating real-world interactions, it helps uncover defects and ensures the software meets requirements. Test scenarios aid in identifying issues early, allowing timely resolution before release, contributing to a reliable software product.

86. What is GUI testing?

GUI (Graphical User Interface) testing is a software testing technique focused on evaluating the visual and interactive aspects of a software application. It involves verifying that the user interface elements, such as buttons, menus, forms, and windows, function correctly and display accurately. GUI testing ensures that user interactions with the application are smooth, intuitive, and aligned with design specifications. Testers assess aspects like layout, font, color schemes, responsiveness, and alignment of graphical elements. GUI testing is essential to guarantee a positive user experience and identify any discrepancies between the intended design and the actual appearance and behavior of the application. By addressing issues early in the development cycle, GUI testing helps enhance usability, user satisfaction, and the overall quality of the software product.

87. When should testing end?

Software testing should end when the predetermined testing objectives have been achieved and the software meets the specified quality criteria. Testing concludes when all test cases have been executed, defects have been identified and resolved, and the software functions as intended. However, it’s essential to consider factors such as time constraints, budget, and the

acceptable level of risk. While complete elimination of defects is improbable, testing should halt when the benefits of additional testing no longer justify the resources expended. A balance between thorough testing and timely release is crucial, ensuring that the software is stable, functional, and aligns with user expectations.

88. What are the different types of Software testing?

Software testing encompasses various types aimed at ensuring the quality, functionality, and reliability of software applications. Unit testing verifies individual components in isolation. Integration testing assesses interactions between integrated components. Functional testing validates if the software functions as specified. Regression testing ensures new changes don't adversely affect existing functionalities. Performance testing evaluates the system's responsiveness and scalability under different conditions. Security testing examines vulnerabilities and safeguards against threats. User Acceptance Testing (UAT) involves end-users validating software suitability. Compatibility testing checks software behavior across different platforms. Usability testing assesses user-friendliness and overall user experience. Exploratory testing involves dynamic, ad hoc testing. Automation testing uses scripts to execute tests. Load testing gauges the application's response under heavy load. Each type targets specific aspects, collectively ensuring a comprehensive evaluation of the software's capabilities and performance before deployment.

89. What is Acceptance testing?

Acceptance testing is a critical phase in software testing where a software application is evaluated to determine whether it meets the specified requirements and is ready for deployment. It involves validating that the software aligns with user expectations, functions as intended, and satisfies the defined criteria for acceptance.

There are two main categories of acceptance testing:

1. User Acceptance Testing (UAT): In UAT, end-users or stakeholders perform tests to ensure that the software meets their needs and requirements. This testing phase confirms that the software is user-friendly, fits into the users' workflows, and provides the desired functionalities.
2. Business Acceptance Testing (BAT): BAT focuses on validating that the software fulfills the business objectives and goals. It ensures that the software aligns with the strategic goals of the organization and that it supports the intended business processes.

Acceptance testing acts as the final gate before software release. Its successful completion indicates that the software is ready for deployment to a wider audience. Any issues identified during acceptance testing are addressed, and once the software passes this testing phase, it signifies that it has met the necessary quality standards and is deemed acceptable for production use.

90. Differentiate between bug leakage and bug release

Bug Leakage:

Bug leakage occurs when a defect that was previously reported and fixed reappears in the software after it was thought to be resolved. This usually happens when the testing team fails to identify the recurrence of the bug during retesting or regression testing. It indicates that the bug was not completely fixed or that the testing process might have missed certain scenarios that

trigger the bug. Bug leakage can lead to a decrease in user confidence and dissatisfaction if users encounter the same issue post-release.

Bug Release:

Bug release refers to the presence of defects or bugs in the software that are identified after the software has been released to the users or clients. These bugs were not detected during the testing process and were inadvertently included in the released version. Bug releases can happen due to various reasons, such as incomplete testing, inadequate test coverage, or unforeseen scenarios. They can result in user inconvenience, reputational damage, and the need for urgent patches or updates to fix the released bugs.

91. What is a test script?

A test script is a set of instructions or lines of code written to automate the execution of test cases during software testing. It outlines the sequence of actions that need to be performed by the testing tool or framework to simulate user interactions with the software. Test scripts specify inputs, expected outcomes, and validation criteria for each step, allowing for repeatable and consistent testing. By automating test cases through scripts, testing efficiency is improved, and manual effort is reduced. Test scripts are particularly valuable for regression testing, where repetitive tests need to be executed to ensure that new changes haven't introduced defects.

92. 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.

93.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

94.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

95. 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.

96. 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.

If you face any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

97. 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.
You can even check out the details of Manual Testing with the Manual Testing course.

98. 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

99.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

100. 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

101. 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.

Bug life cycle - Manual testing interview questions - Edureka

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

102. 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

103. 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.

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

1. Positive Testing
2. 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

105. What is your approach towards a severely buggy program? How would you handle it?

When dealing with a severely buggy program, a systematic approach is essential to identify, prioritize, and resolve the issues effectively. Here's how I would handle it:

Bug Triage: Start by categorizing and prioritizing the bugs based on their severity and impact on the program's functionality. Critical bugs affecting core functionalities should be addressed first.

Isolate and Reproduce: Reproduce the bugs in a controlled environment to understand their triggers and conditions. Isolating the bugs helps in diagnosing the root causes.

Root Cause Analysis: Conduct a thorough investigation to identify the root causes of the bugs. This may involve reviewing the code, logs, and system behavior to pinpoint the areas requiring correction.

Fix the Bugs: Develop patches or fixes to address the identified issues. Implement solutions that not only resolve the immediate problems but also consider the potential implications on other parts of the program.
Unit Testing: Test the individual fixes rigorously through unit testing to ensure that they don't introduce new problems or regressions.

106. 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.

107. 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.

108. 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.

109. 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 system's functionality as a whole to ensure 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\

110. 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

111. 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

112.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.

113. 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 program's code error detection and execution 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

114. 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

115. 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.

If you face any challenges with these Manual Testing interview questions, please comment on your problems in the section below.

116. 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.

117. 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

118. What is a cause-effect graph?

A cause-effect graph, also known as a cause-and-effect diagram or Ishikawa diagram, is a visual tool used for problem-solving and root cause analysis. It was developed by Japanese quality control expert Kaoru Ishikawa and is often referred to as the "fishbone diagram" due to its resemblance to a fish skeleton.

The diagram helps identify and explore the potential causes leading to a particular problem or effect. It's particularly useful in understanding the complex relationships between various factors that contribute to an issue. The main components of a cause-effect graph include:

1. Effect: The problem or issue you're trying to address is placed at the head of the diagram.

2. Major Causes: These are the main categories of factors that could contribute to the problem. They are typically categorized into branches stemming from the main horizontal line (the "spine" of the fishbone).

3. Sub-causes: Each major cause is further broken down into sub-causes or specific factors that could be contributing to the problem. These are detailed within the branches of the diagram.

119. 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

120. 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

121. 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).

122. 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.

123. 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.

124.Why is it impossible to test a program thoroughly or 100% bug-free?

It is impossible to build a software product that 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, outputs, and path combinations to test.

125. 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 understand the entire problem and more flexibly explore other angles of tests. 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.

126 What is Automation testing?

Automation Testing uses an automation tool to execute test cases. The main goal of Automation Testing is to reduce the number of test cases to be run manually and not eliminate Manual Testing.

127 When will you automate a test?

Automation is preferred in the following cases
Repetitive Tasks.
Regression Testing
Smoke and Sanity Tests.
Test with multiple data sets.
Testing is not recommended for one-off test cases. Usually, the decision on which test cases to automate is based on the ROI (Return on Investment). The more times the automated test is executed, the better the ROI.

128. When will you not Automate testing?

One should not automate in the following cases
When the Application Under Test changes frequently
One-time test cases
Adhoc – Random Testing
Exploratory Testing
Usability tests that generally need manual intervention to check the test results
Test cases with detailed setup requirements to be done before each execution
Test cases that return unpredicted test results
Exclude unplanned test case

129. What are the points covered while planning the phase of automation?

During the planning phase of automation, things that must be taken into concern are:
Selection of the “right” Automation tool
Selection Automation Framework, if any.
List of in-scope and out-of-scope items for automation.
Test Environment Setup.
Preparing Gantt Chart of Project timelines for test script development & execution.
Identify Test Deliverables.

130. In what condition you can’t use automation testing for the Agile method?

Automation testing is not helpful for agile methods in the following conditions:
When user stories are constantly changing
When an exhaustive level of documentation is required in Agile.
Only suitable for regression tests during agile testing, like continuous integration.
Learn more about Agile Testing.

131. What is a test script?

A test script is a code to perform a set of instructions on an application. It is used to verify whether the application is functioning as per the software requirements.

When you run your script, it gives the test results as a pass or fails, which is determined by whether the application works as per the expectations.

132. How to select a good test automation tool?

Wide Test Environment support
Easy to use
Good debugging facility
Robust object identification
Record and Playback
Supports common programming languages for test script creation, for example, Java
Image testing abilities
Testing of database
Parameterization
Support multiple automation frameworks
Type of support is available for the tools like documentation, tutorials, training, etc
Cost and budget
Good reporting system

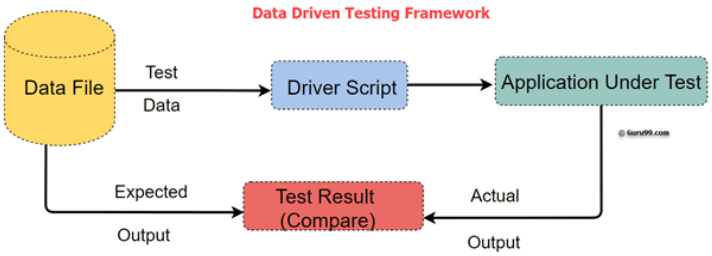
133. What is a Modular Testing framework?

Modular Testing framework is built on the concept of abstraction. In this type of framework, the tester creates scripts for all the application modules under test, and then these scripts are combined in a hierarchical order to create test cases.

134. Data-Driven Testing framework

Data Driven Testing Image

In Data driven testing framework, the input and expected output data corresponding to the input data is stored in a file or database.



The automated script runs the same test steps for multiple data sets. It also allows you to run multiple test cases where only the input data differs, but the steps of execution remain the same.

135. What version control systems do you use?

We use GitHub. Version control helps you to track code changes. It controls the test script source code with a recorded history of changes to simplify the modification process. You may also revert to previous code versions if you make a mistake.

136 What are XPath Axes? Name some of them.

XPath is a syntax that manipulates XML (Extensible Markup Language) data. They help to locate nodes related to those on the tree. Some important XPath Axes are ancestor, child, namespace, parent, etc.

137. How can you speed up an auto test suite?

Applications that require UI testing that interacts with multiple elements can slow down the testing process. It's better to create a simple test script that speeds up test execution.

138. Is documentation necessary in Automation Testing?

Documentation plays a vital role in Test Automation. You should document all the methods and procedures to ensure their repeatability. Test specifications, designs, code changes, test cases, automation plans, bug reports

139. What types of frameworks are used in software automation testing?

Four types of frameworks used are
Data-driven automation framework
Keyword-driven automation framework
Modular automation framework
Hybrid automation framework
Learn more about automation frameworks

140. Is it possible to achieve 100% automation?

No, it is not possible to automate everything. Achieving 100% automation is difficult as there are some scenarios where a registration page has a captcha or some test cases we don't execute often. Moreover, automating these test cases will not add value to the automation or bring positive ROI.

141. What is the average number of test cases you have automated per day?

The answer depends on the length and complexity of the test scenario. Generally, a QA tester can automate 2-4 test scenarios daily when the complexity is limited. However, sometimes it might reduce to 1-2 when the complexity is high.

142. What is the scripting standard while performing automation testing?

While writing the scripts for automation, you must consider the following things:
Uniform naming convention.
3 lines of comments for every 10 lines of code.
Adequate indentation.
Robust error handling and recovery scenario.
Use of Frameworks wherever possible.

143. What are the most popular tools for automation testing?

The most popular test tool for automation testing are:

Selenium

UFT.

Rational Robot.

Here is a complete list of automation testing tools.

144. How can you measure the success of automation testing?

Following criteria can map the success of automation testing:

Defect Detection Ratio

Automation execution time and time savings to release the product

Reduction in Labour & other costs

145. Can you list out some disadvantages of manual testing?

Manual testing requires more time and more resources.

Inaccuracy

Executing the same test case repeatedly is error-prone and tedious.

It is impractical to do manual testing on very large and time-bound projects.

146. What are the differences between open-source tools, vendor tools, & in-house tools in automation testing?

Here are the differences between all:

Open-Source Tools: They are free tools with source code available on the internet. Example: Selenium

Vendor Tools: These testing tools are developed by companies, and you need to purchase their licenses. Example: Microfocus UFT.

In-house Tools: It is built by companies for their use.

147. What are the Prerequisites of Automation Testing?

A few important pre-requisites of Automation Testing are:

A stable build

Functionalities to be tested

Test cases for automated Testing

148 Can you do automation without a framework?

Frameworks are guidelines and not mandatory to create and execute automation scripts. So, yes, we can automate without a framework. Enhancing and maintaining test scripts would be easy if we created and followed a framework.

149. Tell me what you know about Selenium

Selenium is a free (open source) test automation suite. It is used to automate Web and Mobile environments. It consists of the following.

Selenium IDE (Browser Addon–Record and Playback Tool)

Selenium WebDriver
Selenium Grid (Distributed Testing)
Selenium supports scripting in languages like Java, C#, Python, Ruby, PHP, Perl, and JavaScript.

150 Tell me about QTP

QTP (Quick Test Professional) is now known as Microfocus UFT. It is a commercial automation tool and supports an extensive range of test environments: Web, Desktop, SAP, Delphi, Net, ActiveX, Flex, Java, Oracle, Mobile, PeopleSoft, PowerBuilder, Siebel, Stingray, and Visual Basic, amongst others.
The scripting language is VBScript. The tool gels well with ALM (Test Management Tool) and LoadRunner (Performance Testing Tool).
Salient features of QTP include Business Process Testing, keyword-driven framework, XML support, robust checkpoints, and test results.

151. What is SikuliX?

SikuliX is a tool that uses the "Visual Image Match" method to automate the graphical user interface. All the web elements in SikuliX should be taken as an image and stored inside the project.
SikuliX is comprised of
SikuliX Script
Visual Scripting API for Jython
SikuliX IDE
Practical uses of SikuliX are:
It can automate window-based applications and anything you see on screen without using internal API support.
It provides a simple API.
It can be easily linked with tools like Selenium.
Web applications can be automated.
SikuliX offers extensive support to automate flash objects.
It can work on any technology -.NET, Java.

152. Mention what the difference between Selenium and SikuliX is?

SikuliX Selenium
It provides extensive support to automate flash objects It cannot automate flash objects like video players or audio players.
It has a simple API It has got complicated API
It uses a visual match to find elements on the screen. So, we can automate anything we see on the screen. It uses CSS, ID, locators, and other selected to identify GUI elements
It can automate the web as well as windows application It can automate only web applications

153. What are the attributes of a good automation framework?

Here are some important attributes of a good automation framework:
Modular: It is a framework that should be adaptable to change. So that testers should be able to modify the scripts as per the environment.

Reusable: It should be reusable so that methods or utilities should be written in a common file accessible to all the scripts.
Consistent: It should be written in a consistent format.
Independent: The automation scripts should be written in such a way that they are independent of each other.
Integration: Automation Framework should be developed in such a way that it is easy to integrate with other applications.

154. What is Cross-Browser Testing?

It is a subset of browser automation testing that helps you ensure that the online application operates correctly across different browsers. Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, etc.

155. Which Testing can be done using the Selenium Framework?

You can use a Selenium framework for the following testing:
Load testing of web applications.
Regression testing of web applications.
Functional testing of web applications.

156. Is Automation testing white box testing or black box testing?

Automation testing is primarily black box testing.

157. What keyword is used to fetch the URL of the current page in Selenium?

Selenium WebDriver can help you find the current URL of a page with the getcurrentURL(). This method will find the URL of the open applications and result in a string.

158. Where will you maintain information like URL, login, and password?

URL, login, and password are important information used very often and change frequently. They should always be maintained in a separate file. If not done, then the automation tester must change it in every file with its reference.
Automation Testing Interview Questions for 3 to 5 Years Experience

159. What are the Extensions and Test Assets of QTP?

Some Important Test Assets and extensions of QTP are:

Results .xml

Recovery scenario .qrs

Test batch runner .mtb

Shared object repository .tsr

Local object repository .mtr

Test file .mts

Function library .qfl

160. What are the essential modules of an automation testing framework?

Here are some essential modules of the automation testing framework:

Test Assertion Tool: This testing tool will provide assert statements for assessing the expected values in the application under test. For Example, Junit, TestNG, Junit, etc.

Data Setup: Ensures that each test case takes the test data from the database, a file, or embedded in the test script.

Build Management Tool: The framework requires to be built and deployed to create test scripts.

Continuous integration tool: They are required to integrate and deploy the changes done in the framework at each iteration.

Reporting tool: It helps to generate a readable report after the test cases for a better view of the steps, failures, and results.

Logging tool: They help in better debugging of the error and bugs.

161. What is Cucumber?

Cucumber is an open-source (BDE) behavior-driven development tool. It is used tool for web-based application automation testing and supports languages like Java, Ruby, Ruby, Scala, Groovy, etc. Cucumber reads executable specifications written in plain text and tests the application under test for those specifications.

162. What is Test Complete?

TestComplete is an automated UI testing tool for desktop applications, web, mobile, etc. It offers the flexibility to record a test case on one browser and run it on multiple browsers, thus supporting cross browsers testing.

163. What is Cypress?

Cypress is an open-source testing framework. It is developed in JavaScript and has lately gained popularity because of its simplicity and extensive capabilities that enable browser testing, and user manuals should be thoroughly documented.

164. How can you handle the alert popups in Selenium WebDriver?

Selenium gives alerts if there are issues while you test. The pop-up interface allows you to handle the alert by switching the control to the pop-up, pressing the OK or Cancel buttons, and turning back to the source page screen.

String srcPage = driver.getWindowHandle();

Alert pop = driver.switchTo().alert(); // shift control to the alert pop-up.

Pop.accept(); // click k button.

165. What is a Hybrid Testing framework?

The Hybrid Testing framework develops the test cases from modular scripts by combining them in the modular testing framework.

166. Write steps to automate primary "login" functionality test cases for an application?

Here are the steps to automate basic login functionality:

Step 1) Understand the project requirement.

Step 2) Identify the Test scenarios

Step 3) Prepare a data input file with the data corresponding to each scenario

Step 4) Launch the tool from the program.

Step 5) Identify the username, password, and login buttons.

Step 6) Verify that the error message for negative scenarios is the same as the success message for positive test -scenarios.

167. What are the steps involved in the Automation Process?

In the automation process, the steps involved are

Selecting the Test tool

Define the scope of automation

Planning, design, and development

Test execution

Maintenance

168. Can you tell me some good coding practices while automation?

Here are good automation practices:

Add appropriate comments to explain that coding part.

You should identify the reusable methods and write them in a separate file.

Must follow the language-specific coding conventions.

Store the test data in a separate file.

Run your scripts regularly.