**1. What are the different types of manual testing?**

Manual testing has three types of testing: white-box, black-box and grey-box testing.

* White-Box Testing: It is an approach that allows testers to inspect and verify the internal implementation of the software.
* Black-Box Testing: It is an approach that allows testers to test without having any knowledge of the internal implementation. It is just to check whether it is meeting the customer’s requirements without any knowledge of the code.
* Grey-Box Testing: In this approach, we test the software with some knowledge of the code. It is a blend of both white-box and black-box testing.

**2. How do you explain STLC?**

STLC stands for software testing life cycle. Software testing life cycle is the whole process of how testing is performed, and what are the phases involved. The testing life cycle consists of six phases:

* Requirement analysis
* Test planning
* Test case development
* Test environment setup
* Test Execution
* Test cycle closure.

So, it starts with the requirements gathering, where a requirement document is shared with the team. After going through the requirement documents, the test planning is done. ***The test plan is a document describing the approach, resources and scheduling of the testing activities.*** Next, we develop test cases for the functional and non-functional specifications in the test case development phase. After this phase, the environment is set feasible to start testing. After the environment is set, test cases are executed and lastly, the closure of the whole life cycle is done and a report is generated.

**3. Can you explain SDLC?**

SDLC stands for Software Development Life Cycle. The development process adopted in every project is according to its aims and goals. There are six phases involved in SDLC.

* Requirements Gathering
* Design
* Development/implementation
* Testing
* Deployment
* Maintenance

SDLC starts with the requirements gathering, then the design is implemented as per the requirement analysis. After the design is ready, it’s been given to the development team for implementation. After implementation, the product goes into the testing phase. After the product is successfully tested, it is deployed. After deployment, the product is regularly maintained after the user starts using it.

**4. Explain the Waterfall model.**

This model is in a linear sequential manner. This development model involves executing each phase and then going into the next phase.

**Merits:**

* Suitable for small and mid-size products with fixed requirements.
* Easy to determine the critical points in each phase.
* Best when there are enough resources.

**Demerits:**

* Testing happens at the end of the project, so the defects tracked are close to the implementation.
* Not suitable for the projects having frequent changes in requirements.
* Resources have to sit idle and wait for the phases to complete. For example, if the project is in the development phase, then testers have to sit idle and wait for the development phase to complete.
* Backtracking is not possible as requirements are constant.

**5. Explain V- Model.**

As the name suggests, the testing and development life cycle run parallel, the one side of V is verification (development) and the other side is validation (verification). Hence this is also called the Verification and Validation model as well. This model is an extension of the waterfall model, the only difference is that the process steps are bent upwards after the coding phase, to form a typical V shape.

**Merits:**

* Designed for small and medium-sized projects.
* Better than the waterfall model as it allows to test the software side by side.
* V-model provides guidance that testing needs to begin as early as possible.
* It has four Test levels- component, integration, system, and acceptance testing.
* Defect tracking is easy.

**Demerits:**

* Lack of flexibility.
* Changing requirements at the later phase could cost high.
* High business and development risk.

**6. What are test cases?**

It is a document that has the steps that have to be executed. This document is planned before the test case execution phase.

**7. How do you explain the test case format?**

A test case format has the following fields:

* Test case ID
* Test case Description
* Severity
* Priority
* Test Data
* Environment
* Build Version
* Steps to execute
* Expected Results
* Actual Results

**8. What is a test plan?**

Test planning is done before the start of testing activities. It is a document specifying the strategy, scope, approach, resources and schedule of the testing activities. It should cover the following details:

* Test strategy
* Test objective
* Exit/suspension criteria
* Resource planning
* Test deliverables

**9. Is it possible to do exhaustive testing? How much testing is sufficient.**

It is impossible to test everything, instead, we can focus on prioritizing test cases. An extensive test that finds more bugs doesn’t mean we have discovered every defect. Since exhaustive testing is not practical, our best approach as a tester is to pick those test cases that are most likely to find bugs.

**10. When should we start doing testing?**

We should start testing as early as possible. The earlier a bug is tracked, the better it will be for the software. A bug tracked early can save money and time as well.

**11. What is the difference between Quality Assurance (QA) and Quality Control (QC)?**

**Quality assurance** involves process-oriented activities. It ensures the prevention of defects while developing software applications.

**Quality Control** involves product-oriented activities. It executes code to identify the defects in the software application.

**12. Can you explain Bug Life Cycle?**

Bug life cycle mainly refers to the entire phases of detecting a new bug/defect by a tester to closing that bug by the tester. Below diagram describes the stages for a bug life cycle:

In the first stage, a **new** bug is detected by the tester. The bug is documented by the tester and the tester shares the document with the development team. The project manager will then assign the bug to the development team and it comes under the “**assigned**” state. When the development team starts working on the defect, it is in an “**open**” state. In the open state, the bug is classified into different states, duplicate, rejected, deferred, not a bug.

Let’s discuss these states of bugs that developers can assign:

* **Rejected**: If the defect is not considered as a genuine defect by the developer then it is marked as ‘**Rejected’** by the developer.
* **Duplicate**: If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect then the status of the defect is changed to ‘**Duplicate’** by the developer.
* **Deferred**: If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as ‘**Deferred’**.
* **Not a Bug**: If the defect does not have an impact on the functionality of the application then the status of the defect gets changed to ‘**Not a Bug**’.

Next, when the developer fixes the bug and assigns the state to “**fixed**”. After fixing the defect, when the developer shares the code to the tester for retesting, the tester assigns a stage “**Pending Retest**”. When the tester starts retesting, the state is “**Retesting**”. If the bug still persists, the tester assigns it again to the development team and the state is “**Re-open**”. If the tester verifies the bug and finds out it is fixed, then he assigns a state as “**Verified**”. If everything is done and the tester confirms the bug is fixed and he has completed re-testing and the bug no more exists, the bug is assigned to the “**Closed**” state.

**13. What is the difference between Manual testing and Automation testing?**

**Manual Testing** is verifying the software manually and finding defects without the intervention of any tools. The tester has to execute each test case one by one. They manually verify the software as an end-user would. They give input and manually verify the output.

**Automation Testing**is testing the software with the help of some automation tools like Selenium. The code is run for automating tests and the input data is given to the tool. The output data is then compared with the expected results. Automation testing is generally performed for repeated tasks so that the time can be saved.

**14. What do you understand by Unit Testing?**

Testing the smallest piece of code is called unit testing.

**15. What is the difference between severity and priority?**

**Priority** is how soon a bug needs to be fixed.

**Severity**is the impact of a bug on the software.

**16. Give an example of high priority and low severity bug?**

If there’s an issue with the logo of the website, then it will be a high priority and low severity bug.

**17. Give an example of high severity and low priority bug.**

Web page not found when the user clicks on the link, if the user doesn’t visit that page in general. This can be an example of high severity and low priority because the user is not regularly visiting the page so priority is low but the severity is high because its impact is high.

**18. Give an example of high severity and high priority bug?**

An error which occurs on the basic functionality of the application and will not allow the end-user to use the system. This type of bug will be of high priority and high severity. For example, if the user is unable to log in to the application.

**19. What is a traceability matrix?**

Traceability matrix is a document that shows the mapping between test cases and the requirement.

**20. What is the difference between functional and non-functional testing?**

**Functional Testing**focuses mainly on the software’s functional requirements rather than the internal implementation. Functional testing is a type of black-box testing. It mainly focuses on testing the functional requirements which are how the system is behaving? Is it meeting the requirement or not?

**Non-functional Testing**tests the software’s non-functional requirements. Non-functional requirements refer to the quality of the system such as performance, security, scalability and usability. Non-functional testing happens after functional testing. This testing is done to ensure that the software is secure, scalable, high performing and won’t crash under heavy load.

**21. What is the difference between static and dynamic testing?**

Static testing is performed in the early stages of development to avoid errors as it can be fixed easily in the early stages. The errors that can’t be found in dynamic testing will easily be found in static testing.

With dynamic testing methods, the software is executed using a set of input values and its output is then examined and compared to what is expected.

|  |  |
| --- | --- |
| **Static Testing** | **Dynamic Testing** |
| Testing is done without executing the program. | Testing is done by executing the program. |
| This testing does the verification process. | Dynamic testing does the validation process. |
| Performed before compilation. | Performed after compilation. |
| It is about the prevention of defects. | It is about finding and fixing defects. |
| Involves checklist and process to be followed. | Involves test cases for execution. |
| Requires lots of meetings. | Comparatively requires lesser meetings. |
| Return on investment will be high as this process is involved at an early stage. | Return on investment will be high as this process is involved after the development phase. |
| More review comments are highly recommended for good. | More defects are highly recommended for good |
| The cost of finding defects and fixing is low | The cost of finding defects and fixing is high |
| Gives assessment of codes and documentation | Gives bugs/bottlenecks in the software system. |

**22. What is Verification and Validation?**

**Verification**is a static testing technique. Testing is done without executing the code in verification. Examples are reviews, walkthroughs and inspection.

**Validation**is a dynamic testing technique where testing is done by executing the code. Examples are functional and non-functional testing techniques.

In the V model, both these activities go simultaneously.

**23. How can you explain the difference between Black Box Testing and White Box Testing?**

**White-Box Testing** is an approach that allows testers to inspect and verify the internal implementation of the software.

**Black-Box Testing** is an approach that allows testers to test without having any knowledge of the internal implementation. It is just to check whether it is meeting the customer’s requirements without any knowledge of the code.

**24. What is your understanding of Regression Testing?**

**Regression Testing** is performed after re-testing. It is performed on the unchanged part after the bug fixes. In regression testing, software application is tested to make sure that a change is not going to affect the functionality.

**25. What has been your greatest challenges while doing regression testing?**

While doing regression testing the challenges faced are:

* Data Issue. (data is outdated) exmple: if we want to check credit card balance transfer, if balance become ziro after couple transaction then I have to feed the credit card again.
* Code changes, element locator changes etc.
* Compromised business value.
* Time-consuming.
* Large suite to execute.
* Reproducing the defect in production hence becomes inconsistent.
* Re-executing tests repeatedly.
* Improper selection of regression test cases might skip a major regression defect to be found.

**26. What is the difference between smoke testing and sanity testing?**

**Sanity testing**is a kind of software testing performed after receiving a software build, with minor changes in the functionality (code). This is done to ensure that the bugs have been fixed and no further issues have been introduced due to these changes.

**Smoke testing**is a special type of testing performed on software build to check the critical functionalities of the program.

**27. What is the difference between System Testing and User Acceptance Testing (UAT)?**

**User Acceptance Testing (UAT**) is the last step before the product goes live or before the delivery of the product is accepted. UAT is done after the product itself is thoroughly tested.

**System** **Testing** is the level of testing where complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements.

**28. What do you mean by integration testing.**

**Integration testing**is testing where individual modules are combined and tested. It is a level of testing, usually performed after unit testing.

**29. What is the difference between alpha and beta testing?**

**Alpha Testing**is a part of user acceptance testing and is performed internally before delivering to the client. It is performed to ensure no bugs present at the time of delivery.

**Beta Testing**is performed by the customers at their end in a real production environment**.**This is done to get real feedback from the users using the software. It is also a part of user acceptance testing.

**30. Differentiate between ad-hoc testing and exploratory testing.**

**Ad-hoc testing**includes learning the application first and then proceeding with the testing process.

**Exploratory testing**is a form of testing that involves learning of the application while testing.

**31. What are static testing techniques?**

**Static** **testing** is the testing of the software work products manually, or with a set of tools, but they are not executed. It starts early in the life cycle and so it is done during the verification processes. It does not need a computer as the testing of the program is done without executing the program, e.g., inspection, reviewing, walkthroughs.

There are two types of static testing: **Reviews** and **Static Analysis** by tools.

**Reviews** are **informal** and **formal. Informal** **reviews** are applied during the early stages of the life cycle of a document. A two-person team can conduct an informal review, as the author can ask a colleague to review a document or code. In later stages, these reviews often involve more people and a meeting. The goal is to help the author and to improve the quality of the document. Informal reviews have types but there is one characteristic that is common in all informal reviews is that they are not documented. The **formal review** process is related to the factors like the maturity of the development process, any legal or regulatory requirements or the need for an audit trail. Inspection is the most documented and formal review, yet it is not the only one.

**Static analysis** is just another form of testing. It is focused on finding defects rather than failure and the goal is to find the defects whether or not they may cause failure.

**32. What is the difference between formal and informal reviews.**

*An****informal review****is a review not based on a formal(documented) procedure.*

*And,****Formal review****is a review characterized by documented procedures and requirements, e.g., inspection.*

Informal reviews are not documented and performed at early stages. Whereas formal reviews are always documented. Inspection is the most documented and formal review, yet it is not the only one. Below are the differences listed between formal and informal reviews:

|  |  |
| --- | --- |
| **INFORMAL REVIEW** | **FORMAL REVIEW** |
| Conducted on as needed i.e. informal agenda. | Conducted at the end of each life cycle phase i.e. formal agenda. |
| The date and time of the agenda for the informal review will not be addressed in the project plan. | The agenda for the formal review must be addressed in the project plan. |
| The developer chooses a review panel and provides and/or presents the material to be reviewed. | The acquirer of the software appoints the formal review panel or board, who may make or affect a go/no-go decision to proceed to the next step of the life cycle. |
| The material may be as informal as a computer listing or hand-written documentation. | The material must be well prepared.  For ex- Formal reviews include the software requirement reviews, the software preliminary design review, the software critical design review, and the software test readiness review. |

**33. What is a latent bug.**

A**Latent defect** is a bug that remains dormant and does not cause any failure as the exact set of conditions have never been met.

**34. What do you understand by equivalence partitioning.**

***Equivalence Partitioning****is a black box technique in which test cases are designed to execute representatives from equivalence partitions. In principle, test cases are designed to cover each partition at least once.*

The idea behind the technique is to divide a set of test conditions and each partition is then tested separately. We need to test only one condition from each partition because we are assuming all the test conditions in a partition will be treated by the software in the same way. If one condition in a partition works, we assume all the conditions in that partition will work. Similarly, if one condition in a partition fails, then we will assume the entire partition doesn’t work.

**35. What is boundary value testing.**

***Boundary value analysis****is a black box design technique in which test cases are designed based on boundary values.*

In boundary value analysis, we have both valid boundaries (in valid partitions) and invalid boundaries (in invalid partitions). A boundary value includes minimum and maximum values at inside and outside boundaries. BVA is a part of stress and negative testing.

BVA gives testers ease to create test cases for an input field. Let’s take an example of an address text box that allows a maximum of 500 characters. Writing test cases for each character would be a challenging situation, so we can use boundary value analysis.

**36. How do you select regression test cases or form the regression test suite?**

We will select the regression test cases and form the regression test suite by including the test cases

* That verify core features of the application
* For functionalities that have undergone recent changes

 That have frequently yielded bugs.

**37. Why is impact analysis important?**

To practice Risk-based, Impact analysis has to be done. By doing so test cases can be designed in a way that all the severe bugs, critical from the customer’s view can be solved before time. A good study of the business, client’s need and their usage of the software is to be taken care of.

**For Example,** the most important risk associated with software in the banking domain is Security. Any new form added to the already existing software can be vulnerable. A good amount of security testing is advisable by adding proper links, redirection, and navigation to the proper page, installing proxy if needed.

**38. What do you understand by performance testing?**

**Performance testing**is testing the **stability** and **response** **time** of the application by applying **load**.

**Response** **time** is the time taken to send the request to the server, run the program in the server and take back the request from the server.

And the **load** is the number of users using the application at a particular period of time.

**39. What is the difference between stress testing and load testing?**

**Load Testing**helps us to study the behavior of the application under various loads. The main parameter to focus on here is the response time. Load testing is done to ensure how many concurrent users that server can handle effectively and quickly.

**Stress testing**helps us to test the stability of the application. The main objective of stress testing is to identify the breaking point of the server.

**For example,** if 50 users are active on an application for approximately 3 seconds at the same time, then in **load** testing we will be testing for less than or equal to 50 users in 3 seconds. But in **stress** testing, we will be testing for more than 50 users in 3 seconds.

**Behavioral interview questions and answers**

**40. Have you ever managed to write the test cases without having any documents?**

You have to answer this question very carefully and first, you need to tell which projects you have worked on. Whether these projects were in-house or not? Your answer should be related to your previous or ongoing projects.

For example: In one of my previous projects, we had to redevelop our internal tool with new technology, but there were no tests/documentation for the old/existing product. As there is no documentation, below are the steps I have followed:

* Understand and explore the existing product to come up with scenarios
* Understanding the tool from the product owner or senarios or developers who had worked earlier on the same tool.
* Going through production bugs that are found previously for products so that edge test cases are not missed in writing the tests for the upgraded products.

**41. Suppose you find a bug in production. How would you make sure that the same bug is not introduced again?**

Finding a bug in production can be very challenging. To make sure the same bug is not introduced again, we need to add that uncaught functionality to regression test cases. If we have an automated regression suite, then we need to write a new script that validates the above functionality.

**42. If a small section of code in the application is updated, what is your approach to validate it?**

If any small change is done in the code, a tester needs to discuss with the concerned developer that what areas he has updated. After getting the information, a tester will try to perform testing on the same page. If the page has three links for example and the developer has changed only one link. In this case, the tester needs to test the remaining links in the page and then the page itself. It’s not necessary to test the entire application again.

**43. How would you begin testing the build that you have recently received? Is there any approach to follow?**

Yes, we follow an approach as follows:

Smoke Testing> Exploratory Testing> Functionality Testing> Sanity Testing> Regression Testing and Final Product Validation

**44. What will be your approach if in case you have any doubts regarding the project?**

In case of any doubts, the first approach should be to go through the documentation available. If this would not help, then you should approach your senior team members.

We can also approach the business analysts and the development team. The last option would be to approach the manager and the stakeholders.

**45. How do you solve if there is any conflict with your peer QA on any technical aspect?**

There should be an argument but only up to a certain extent with your peer on why you are correct. If the conflict still persists then involve the team and discuss the conflict issues with larger audiences that may be other QAs, development team and scrum masters. In the meeting, you can have open suggestions and get the right direction. You must accept any decision made from the team meeting with a smile.

**46. What do you do when your developer denies that what you have filed is not a BUG?**

When you file a bug and your developer denies it, then we can take the following measures:

* We can provide business documentation reference to support why the bug is not as per design.
* We can have a meeting with the product owner/business analyst for the discussion regarding the bug and how it is deviating from the requirement document.

If the bug is not reproducible, then,

* Provide screenshots of the bug, give timestamps on when you reproduced this so that developer can check in application logs.
* Provide test data you have used for replicating the issue.

**47. What are the drawbacks of agile implementation/methodology that you faced?**

Agile methodology is very popular because of its flexibility to take testing and development parallel but it has some drawbacks as well.

* Sprints are very deadline-constrained.
* Documentation is not a priority.
* Frequent changes in requirements can be sometimes messy.

**48. What if the software is so buggy it can’t really be tested at all?**

If the software is buggy, first we need to categorize the bugs as per their severity. The critical bugs can impact the software and need to be fixed quickly. For this, you need to let the manager know with proper documentation as proof. 

**49. Explain what will be your reaction if a project you had been working on gets a sudden change in the deadline.**

As a tester, we have to be open to our thoughts if we can deliver the project with QA signoff covering all the test cases. If pre-release is a must, then we need to discuss the opportunities of increasing QA resources or the possibility of partial product delivery. We have the power to hold QA sign off if we are not satisfied with the quality of the product which eventually stops the release date.

**50. Write test cases on any device/ object present around (Example: Chair).**

When asked questions on these scenarios, always start with gathering requirements. Ask questions after selecting your object. This will show your knowledge level in the software development life cycle.

In this case:

* Ask for the **type** of chair, office chair, study table- chair, sofa chair, dining table chair, comfortable chair.
* What **material** it is made of? Wood, steel or plastic.
* Ask for the **dimensions** of the chair such as height, weight based on the type of chair.
* Ask for **availability**.

Start making test cases based on the specifications gathered. Test cases would differ for each type of chair, which is better left for your thinking capability (**For Example,** the purpose of chair, dimensions according to the type of chair, portable-non portable, light-weight, purchase options). For each chair, a **performance test case can be:** to derive the tensile strength or the maximum weight-bearing capacity.