MANUAL TEST INTERVIEW QUESTIONS

1. Definition of SDCL and STLC, the differences and similarities between them and the stages of the two concepts

SDLC: The Software Development Life Cycle is a process used by the software industry to develop, test and design high-quality software.

STLC: The Software Testing Life Cycle is a set of activities performed by the test team to maintain or assure the quality of the software or product.

SDLC and STLC are two very similar processes. However, they differ from each other in terms of the area they cover. These two concepts can be expressed as two separate processes intertwined with each other.

STLC is a part of SDLC processes and while SDLC covers the whole process of developing a software product, STLC covers only the testing phase.

As can be seen in the image below, they both consist of 6 stages.

Stages in SDLC:

1. Requirements Gathering

2. Analysis

3. Design

4. Coding

5. Testing

6. Deployment

Stages in STLC:

1. Analysis of requirements

2. Test Planning

3. Test Design

4. Setup of the Test Environment

5. Test Execution

6. Test Completion

Harish kumar

2. What is a Positive and Negative Test? Explain with an example

Positive tests are <u>happy path</u> scenarios to test that the application performs as expected with the right input.

Negative testing is the practice of testing by entering incorrect data into the system to ensure that the application responds appropriately. With these tests, we can identify the points where the software does not work. With this test, we can test how the software works in negative situations, what it shows in negative results, and the durability of the unit.

The most common and well-known example here is an application login module. It is possible and necessary to perform both positive and negative tests on the login module, which is one of the vital modules of the application and has a high priority. We test whether the application works correctly with valid data by writing positive test cases with the boundary value analysis method in the context of the requirements given to us. In the same way, it is also necessary to test the application with different data other than the requirements. For example, by entering invalid values, we can perform negative tests and test the quality of the application. This is also a negative test application.

3. What is a Bug, what are the Definitions of a Bug and what is the difference between a Bug and a Defect?

Bug is the informal name for defects, which means that the software or application does not work according to the requirement.

Terms	Description	Person Responsible
Defect	A situation that occurs when the application does not work according to the requirements. It is the name given to errors found in a project that has not yet gone live.	Test Engineer
Bug	It is the informal definition of a bug. We use this definition to describe errors that we encounter in a live application. In this context, this is the main difference with defetc.	Test Engineer
Error	An error caused by a problem in the code.	Developer and Test Engineer
Issue	The application is the situation that arises from the business requirement.	Customer
Mistake	It is the concept that defines the problems in the document.	
Failure	Identifies a large number of errors that cause the software to fail.	

4. Can you explain the Error Management Process and Error Report Content?

A Bug Report in Software Testing is a detailed document about bugs found in a software application. The bug report contains every detail about the bugs such as description, date the bug was found, name of the tester who found the bug, name of the developer who fixed the bug, etc. The bug report helps to detect similar bugs in the future so that they can be prevented.

When reporting the bug to the developer, your Bug Report should contain the following information;

Defect_ID - Unique identification number for the defect.

Error Description - Detailed description of the error, including information about the module where the error was found.

Version - The version of the application where the error was found.

Steps - Detailed steps with screenshots where the developer can reproduce the errors.

Date Created - Date the error was created

Reference - Here you provide references to documents such as requirements, design, architecture or even screenshots of the bug to help understand the bug.

Detector - Name/ID of the tester who reported the error

Situation - State of the defect

Corrector - Name/ID of the correcting developer

Date closed - Date the error was closed

NOTE: Severity, which describes the impact of the defect on the application, is the priority related to the urgency of defect remediation. The severity priority can be **High/Medium/Low** respectively, depending on the urgency of the impact of the defect that needs to be fixed.

5. Explain the concepts of testing and debugging and state their differences.

Both are used to find bugs in a system/software/application/tool and then fix them appropriately. However, there are several differences between testing and debugging. Testing is the process where we find bugs and errors. **Debugging is the process where we fix the bugs we find in the testing process.**

What is a Test?

Basically, it is a process by which we verify and validate that an application or software is bug-free, meets all technical requirements **(it is not possible to be one hundred percent bug-free)**, complies with all development and design requirements, and meets all user requirements. Testing ensures that the intended software or application meets these requirements efficiently and effectively and addresses all edge cases and exceptions.

What is Debugging?

Basically, it is a process where we fix any bug found in a software or application. In this process, we first detect the bugs, then analyze them and then fix them. Debugging starts after the intended software fails to execute properly. Here, we conclude by solving the problem and testing the software successfully. This process is considered extremely tedious and complex. because at each stage of debugging we need to identify and resolve existing bugs.

6. How does a bug or defect occur?

A bug or defect in software testing can be caused by incorrect, faulty or excessive coding. Incorrect coding is actually building the wrong application. To give an example in this context:

Let's assume that when we click on the "Inbox" link in the Gmail application, it goes to the "Draft" page, this is due to incorrect coding by the developer and is therefore a bug.

Incomplete coding again means that the requirements are not met. In this direction: with incomplete coding, an application that should be done in the context of the example we just mentioned will not actually be done.

Requirements are very important for both a developer and a tester. For this reason, one should **not take one step more or one step less than the requirements.** Excess code can cause the desired application to go wrong in this context.

7. Mention the Importance of Error Tracing Tools and Identify Some of the Commonly Used Error Tracing Tools in this Context.

1. Improving software quality is one of the most basic needs for companies. In this context, a bug tracking tool ensures that the software is released with minimal problems. Bug tracking tools enable early detection of problems, troubleshooting and correction of bugs. It also analyzes what is done by testers to fix a bug or issue. This ensures that an efficient product is delivered on time within the allocated budget and planned time.

2. Bug tracking tools minimize problems caused by human error. Of course, there are still manual entries to these tracking tools, but since **this is the bug memory**, it makes it easier to track defects that may be overlooked due to forgetting and other reasons. Seeing recurring defects provides many benefits, such as easier and more systematic tracking of the tester's work and less time wasted on trivial issues. This saves time and money.

3. Since bug tracking tools are also suitable for teams to work together, they are tools that strengthen communication and enable problems to be solved very quickly in a common way from a center. The resulting report provides the necessary data for planning the next project.

4. One of the biggest advantages is that records of the bug fixing process are securely saved in a centralized data system. This means that even after fixing the bug, you have a detailed list of the bugs, the fixing process and how long it actually took to fix a particular issue. Basically, the entire testing phase is recorded in the central data system, which allows you to keep track of bug trends.

5. It helps testers detect problems much earlier and take action on a priority basis. Human tracking, on the other hand, cannot achieve this in a short time. As a result, the probability of software being released with functional glitches is almost zero when you run it with a bug tracking tool.

-- Some of the commonly used bug tracking tools are as follows:

- **Jira**
- **Bugzilla**
- **BugNet**
- **Redmine**
- **Mantis**
- **Trac**
- **Backlog**
- **Confluence**

8. Briefly Describe the Test Levels in a Software Test

-- Unit Testing: This test aims to verify each module of the software individually. Therefore, these tests are performed to ensure that these verified modules meet the acceptance criteria and provide the expected functionality. **[Performed by the developer] -- Testers who know the internal structure of the code can also perform this test.**

--Integration Testing: It is a test level to test whether the units that interact with each other work together and in harmony with each other. Thus, errors that are overlooked with unit testing can be found at this test level. **(It is performed by developers on a unit basis and by testers on a system basis)**

--System Testing: This is the test level where all components of the software are tested as a whole (**performed by testers).**

--Acceptance Testing: Acceptance Testing is the step in the software testing process where a product is approved or rejected. The purpose of this type of testing is to verify whether the system meets end-user requirements and is ready for deployment. (**Performed by users**)

9. What are the Differences Between Test Case and Test Scenario?

A test case defines each feature that can be tested. It is also called Test condition or Test possibility. Testing the login feature is an example of a test case.

A Test Case aims to test a specific feature of the software more specifically. For example, in a test case for logging in, entering valid values for logging in is the subject of the test case. In other words, test case is a narrower definition while test case is a more general concept. There can be many test cases in a test case. Test cases can consist of positive and negative cases. In other words, these are done to measure that the application reacts correctly by testing the desired and undesired situations.

10. What are the Differences Between Functional and Non-Functional Tests?

Functional Testing: It is a type of test that ensures that each function of the software works in accordance with the specification (specified requirements). These tests test what the system does.

Non-functional Testing: A type of testing used to evaluate the non-functional characteristics of a software application (such as performance, usability and reliability). It evaluates the performance of the system.

11. What are the Differences Between Verification and Validation?

-- *Are you building it right?* This question describes the concept of Verification.
-- *Are you building the right thing?* This question describes the concept of Validation.

What is Verification?

In software processes, software developers take the role of verification. It is a process to verify whether the output conforms to the specifications specified in the product requirements. In other words, it is a set of operations in which the software developer checks whether the code output is in compliance with the standards and requirements specified in the document. In software processes, verification is usually performed first. By performing the verification process, a situation that is overlooked in the validation process to be performed later can be detected.

What is Validation?

In software processes, testers play the role of validation. **It is to check whether** the output or product **meets the customer's expectations.** In other words, it is a set of operations in which the tester compares the code coming from the software developer with the actual

product that the customer wants and checks whether the correct product is produced. In software processes, validation is usually done as the second process after verification. Validation can detect a situation that was overlooked in the previous verification process.

12. When Do We Start Testing in a Project and What are the Benefits?

Software testing should start early in the Software Development Life Cycle (SDLC). This is called shifting left. The aim is to find bugs in the software as early as possible. This helps to identify and eliminate problems during the requirements gathering and design phases of the SDLC. Starting testing early reduces the amount of defects and ultimately the cost of rework. According to one of the seven principles of software testing, "Early testing saves time and money."

13. How Do We Test If User Requirements Are Not Clearly Written?

1. First of all, I try to use all the documents available to me to do the test properly.

2. I can run tests using the version of a similar application and an older version of an existing program as a reference. Thus, I can create my own test data by making use of similar applications.

3. I talk to the project team members about the implementation and take their ideas into consideration.

4. If I have enough experience, I can do exploratory testing. At the end of all this, I will have a project document.

14. What is Exploratory Testing, Under What Circumstances Is It Conducted and What Is Its Scope?

Exploratory Testing is a type of software testing where test cases are not created in advance and testers check the system at the time of testing. It is based on freely exploring the software without being bound to any test case and reporting the bugs they find. It is a hands-on approach where testers participate in minimum planning and maximum test execution. They can jot down ideas on what to test before the test execution.

A type of software testing approach where test design and test execution are carried out simultaneously. Exploratory testing is done when requirements are few or vague and there is pressure on time.

It is within the scope of this test to measure whether the system fulfills the basic functions it needs to perform, especially the places that need to be tested are vitally important and whether the application gives the same response with tests on different devices. In this

context, it is possible to test whether there is a crash in the most used places in the system, whether the system can be logged in with invalid data, testing the new version to be released, and the reaction of the application on different browsers and devices with exploratory testing. For this reason, this type of test is applied randomly and without any configuration.

15. Explain the difference between confirmation and regression tests.

- Confirmation tests: After a bug has been resolved, all test cases that failed due to the bug can be retested; these tests must be rerun on the new version of the software. New tests can also be written to test the software in case the bug was caused by a lack of functionality. At a minimum, the test steps required to reproduce the fault(s) or defect(s) caused by the defect must be re-run on the new version of the software. **The purpose of a confirmation test is to confirm whether the original defect has been successfully resolved.**

- Regression tests: It is possible that a change to one part of the code (be it a fix or another type of change) may inadvertently adversely affect the behavior of other parts of the code (within the same unit, in other units of the same system, or even in other systems). Changes can also be changes to the environment, such as a new version of the operating system or database management system. These unwanted side effects are called regressions. **Regression testing involves testing to find such unwanted side effects.**

Note: Confirmation tests and regression tests can be performed at all test levels (unit, integration, system and acceptance).

16. What are the Differences Between Black-box, White-box and Grey-box Tests?

Black-box: An approach to software testing in which the tester is not aware of the internal structure of the object under test and focuses on the functions of the system.

White-box: a software testing approach where the tester is aware of the internal structure (code, etc.) of the object under test. Unit testing, which is a sub-category of white-box testing, is mostly performed by developers. However, QA's are also capable of performing this test if the necessary source code is provided. Unit Testing is the independent testing of all the functions we do one by one. It precedes integration testing and is the first level of software testing.

Grey-box: It is a type of test that means the combination of knowing the source code required to perform the test in white-box testing and black-box testing that is not required. This type of test is performed by the tester and reduces the burden of the developer.

However, in order to perform this test, the tester must have the source code developed by the developer and have the skills to use it. A tester has the competence to perform this type of test when he has the necessary data. In summary, this test tests both the internal requirements of the system and the compatibility of inputs and outputs. The purpose of gray box testing is to look for and detect errors caused by poor code structure or application usage.

17. What is Static Testing, What are the Benefits and What are the Differences with Dynamic Testing?

Definition: A type of testing based on requirements analysis prior to execution of the source code. It is based on manual inspection of the software or other work products, or evaluation of the code or other work products using tools (static analysis). This test checks for logical inconsistencies, sentence fragments, incompatibilities, spelling, grammar and readability.

Benefit: Static testing can find faults early and reduce costs. Errors that cannot be found in dynamic testing can be found at this stage and an efficient system can be obtained. Errors in design and coding are eliminated. **Thanks to static testing, a sustainable code structure is obtained.**

Scope: When static testing, we not only analyze the requirements, but also look for inaccuracies in the code, for example, if there are variables that are not assigned a value without running the code, if the code is unnecessarily repeated, if there is unreachable code, or if the code is not used even though the value is assigned, it is possible to find this with static testing. We can find these situations expressed with a manual review. Apart from this, it is also possible to find design errors, security vulnerabilities and incomplete test writing in the context of acceptance criteria with static testing.

18. What do you know about the definition, content and benefits of a Test Plan?

Description and Content: A document describing the scope, strategy, resources and timeline of planned test operations. It specifies, among other things, the test items, the features to be tested, the test tasks, who will perform each task, the degree of independence of the tester, the test environment and the test procedure.

The design methodologies and entry and exit criteria to be used, as well as the rationale for their selection and the hazards requiring contingency planning, are a record of the test

preparation procedure. As the project and test preparation continues, more information may become available and more detail may be added to the test plan. Test planning is an ongoing process throughout the life cycle of the product.

Benefit: Helps people outside QA teams (developers, business managers, customer-facing teams) understand exactly how to test the website or application. They provide a clear guideline for QA engineers to conduct testing activities. They detail test coverage, test estimation, strategy, etc. Having all this information in a single document makes it easy to be reviewed by management personnel or reused for other projects.

19. What is the Content of Test Summary and Test Progress Reports?

The purpose of test reporting is to synthesize and share information about test activities, both during test activities and after their completion (for example, a test level). A test report generated during a test activity is known as a test progress report, while a test report generated at the end of a test activity is known as a test summary report.

- The content of the test report varies depending on the project environment and the intended audience of the report.

20. What Do You Think of When You Think of Risk-Based Testing?

First of all, it is necessary to define risk. In this context, the probability of an event or situation that will cause negative consequences in the future is called risk. This type of test is software testing based on the probability of risk. In some cases, the software may be complex, it may be difficult to make it business critical, or potential problems may occur due to the frequency of use of some features in the product and the sensitivity of the product. These and similar situations need to be analyzed and necessary precautions need to be taken. Accordingly, risk-based testing prioritizes the testing of more important and potentially defective aspects and functions of the software.

In terms of its contribution to the project, we can say the following: It allows problems to be recognized as early as possible, tests to be run with fewer problems, minimizing possible risks and contributing to the timely completion of the project.

Risk-based testing aims to determine the test techniques to be used, the levels and types of tests to be run, the scope of the tests to be run, the prioritization of tests to identify critical defects as early as possible, and the potential activities that can be done in addition to the tests to reduce risk.

21. Give Information about Alpha and Beta Tests and Explain Their Differences

Alpha and beta testing are often used by developers **of commercial packaged software** (COTS) who want to get feedback from potential or existing users before the product is released.

Alpha tests are performed in the software developer's own environment, not only by the software development team, but also by potential-existing customers or an independent test team. **Beta tests**, on the other hand, are conducted by potential-existing customers or operators in their own environments. Alpha tests can be performed before Beta tests or only beta tests can be performed without any alpha tests.

One of the objectives of alpha and beta testing is to build user confidence that the system can be used under normal conditions and in operational environments with minimal difficulty, cost and risk. Another objective is to identify bugs related to the conditions and environments in which the system will be used.

22. What Mistakes Testers Tend to Make.

1. Don't be afraid to communicate,

2. Fear or hesitation to ask questions,

3. Don't start testing without fully understanding the scope and requirements,

4. Generating error reports with insufficient content,

5. Ignoring some requirements when writing test cases,

6. Don't start working without any planning,

7. False positives and negatives.

23. What are the Benefits and Advantages of Test Independence?

-Because of their different experiences, technical perspectives and biases, independent testers are more likely to recognize different types of defects than developers.

-An independent tester can verify, challenge or refute assumptions made by stakeholders during system specification and implementation.

-A supplier's independent testers can report honestly and objectively on the system under test without (political) pressure from the company that hired them.

24. Which topics are essential in a Bug Report?

1. Title

2. Steps to reproduce

3. Expected result

4. Actual result

5.Priority

6. Screenshot or video

25. What are the potential disadvantages of test independence?

-Isolation from the development team can lead to a lack of cooperation between the two parties, delays in providing input to the development team, or an adversarial relationship with the development team.

- Developers can lose a sense of responsibility for quality.

- Independent testers can be seen as an obstacle by developers.

26. What are the differences between testing techniques and testing tools?

The purpose of any testing technique is to help define test conditions, scenarios and data.

- Black Box Techniques (Equivalence Partitioning)

- White-box Methods (Expression Coverage)

- experience-based testing techniques (Error estimation)

- In the context of software testing, a tool is a product that supports one or more testing activities such as planning, requirements, build creation, test execution, defect tracking and test analysis.

Test Management Tools (Google Sheets-Trello-Jira)

Test Automation Tools (Selenium Webdriver - Cypress - Robot Framework)

Performance Testing Tools (Jmeter - HP Loadrunner)

API Testing Tools (Postman - Soap UI - Rest Assured)

27. What are the Differences Between Equivalence Partitioning and Boundary Value Analysis Testing Techniques?

Equivalence class means that all members of a given partition are treated in the same way, i.e. the data is divided into partitions, i.e. equivalence classes, as would be expected.

- Boundary value analysis (a continuation of the equivalence class. It can only be used when the partition is ordered and consists of numerical or ordered data) The boundary values of a partition are its minimum and maximum values (or initial and final values).
- Behaviors near the boundaries of the equivalence class are more likely to be incorrect than behaviors inside the partitions.

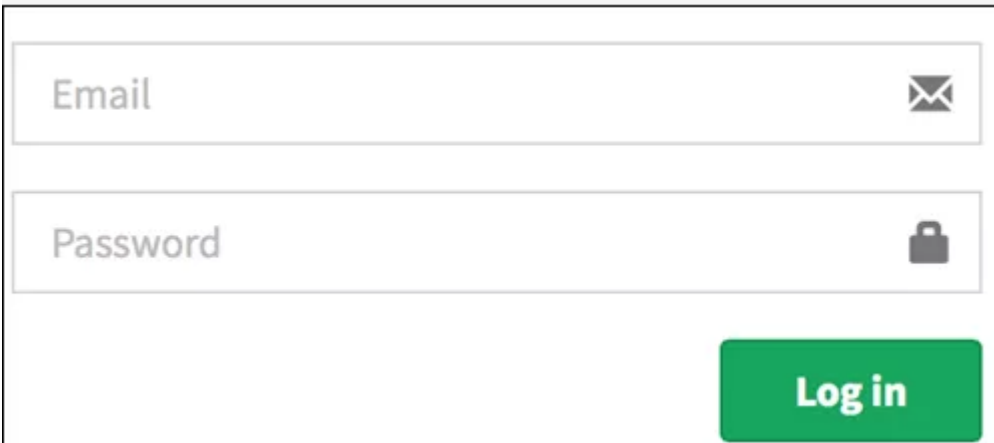### 28. What is Random/Monkey Testing and When Do We Need It?

Another name for random testing is monkey testing. In this type of testing, data is generated randomly, usually using a tool or an automated system. The system is tested with this randomly generated input and the results are evaluated accordingly. These tests are less reliable. They are therefore usually performed by beginners to determine whether the system can withstand bad results.

29. What is Decision Table Testing and When Do We Use It?

Decision table testing is a testing technique used to test systems whose properties are in the form of rules or cause-and-effect combinations. In a decision table, inputs are listed in a column and outcomes are listed in the same column but below the inputs. In the rest of the table, combinations of inputs are examined to identify the outputs produced. It allows us to get better test coverage. We can also call it a **Cause and Effect table**.

-- Let's see what is meant by an example table:

Let's imagine a system where we log in by entering a username and password.

We can draw a decision table as follows;

CONDITIONS	RULE 1	RULE 2	RULE 3	RULE 4
User Name (T/F)	F	T	F	T

Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

* T= TRUE, F= FALSE, E= YES, H= NO

30. What are the Differences Between Negative and Positive Testing? Can you give an example from your project?

Negative test cases are used to check how the software will behave in unexpected situations. These tests can include unexpected situations such as user-entered data that is incorrect, system failures, users who are not authorized to access system resources. Examples include scenarios such as a user trying to access an account by entering the wrong password or an application using an incorrect URL to connect to a database.

Positive test cases are used to verify that the software works correctly. These tests are designed to check the expected behavior of the software. Examples include scenarios such as a user trying to access an account using the correct password or an application connecting to a database using the correct URL.

For an example project, let's consider the test cases of an e-commerce web application:

Negative test scenarios:

- User attempts to log in to their account with an incorrect password
- The user adds an out-of-stock product to the cart before proceeding to checkout
- The user enters incomplete or incorrect credit card information during the payment process

Positive test scenarios:

- The user logs into their account with the correct password
- The user adds a product to the cart, proceeds to the checkout page and makes a successful payment
- The user can track the order status and receive the right product at the right time

31. What Do You Know About the Waterfall Model?

The Waterfall model is a frequently used method in the software development process. It is a method in which the entire process is linear, working in the order in which each phase must be completed.

In this model, the process consists of the following stages:

Requirements Analysis: In this phase, customers' demands and needs are collected and analyzed. The required system features, functions and technologies to be used are determined.

Design: In this phase, the system architecture is created in line with the requirements determined in the previous phase. In this phase, issues such as database structure, user interface, data flow and functionality are determined.

Development: In this phase, software coding is performed and system functionality is created.

Testing: This is the phase where software testing will be performed. In this phase, the software is tested to make sure that it fulfills its desired features and works stably.

Implementation: Applying the software to the target system and making it ready for use.

Maintenance: This phase includes fixes and improvements to optimize the functionality and performance of the software.

32. What are the Activities in the Maintenance Phase, the Last Phase of the SDLC?

1. Troubleshooting: During the maintenance phase, errors and problems that arise during the use of the software are fixed. This is important to ensure that the software works correctly.

2. Change Management: The maintenance process is used to manage changes to the software. These changes can be based on factors such as customer feedback, security vulnerabilities or new business requirements.

3. Update: The maintenance process includes updates to keep the software up to date. These updates can be made based on factors such as new technologies, business requirements or security vulnerabilities.

4. Documentation Update: The maintenance process is used to update the documentation of the software. This can include user manuals, operating instructions and help files.

The maintenance process is important to provide the best experience to the users of the software. Maintaining the software helps to minimize problems that users may encounter when using the software and improves the functionality and performance of the software.

33. Can you explain the differences between Waterfall and Agile?

Waterfall and Agile are two different methods in the software development process. Waterfall is an old method and has been used traditionally, but Agile is a newer approach and is suitable for more modern software development processes.

Waterfall is a method in which the software development process follows a linear path. This means that each phase must be completed and the next phase cannot start. So, once one phase is completed, the next phase is started. For example, requirements analysis follows

certain phases, such as design, coding, testing and release phases. This method is often used in large-scale projects.

Agile, on the other hand, is a method where the software development process is more flexible and dynamic. In this method, all phases are performed simultaneously and the software development process continues in a cycle. Each cycle is improved based on the knowledge learned from the previous cycle. The Agile method is more commonly used in small and medium-sized projects.

The main differences between the two methods are as follows:

While the Waterfall method follows the phases in a specific order, the Agile method performs the phases simultaneously.

While the Waterfall method is less flexible and requires long-term planning for projects, the Agile method is more flexible and can be changed quickly according to the needs of projects.

The Waterfall method is difficult to go back as one phase must be completed before moving on to the next phase, but the Agile method is reversible and mistakes and misunderstandings made in each cycle can be corrected.

The Waterfall method requires more time to handle changes, while the Agile method is suitable for making changes faster.

To summarize, the Waterfall method is a traditional approach, while the Agile method is a more recent and modern approach. Both methods offer different advantages for different projects. The Waterfall method is more suitable for large-scale projects, while the Agile method is more suitable for small and medium-sized projects.

34. Which Points Should We Pay Attention To Bring Out The Best Test Case?

- I write test cases from the end user's perspective
- I write the test steps in a simple way that everyone can easily follow
- I design test cases in a reusable format.
- Priority is extremely important. In this context, I pay attention to priority.
- I include a test case description, test data, expected result, precondition, postcondition data in my test case.
- I write valid test cases as well as invalid test cases.
- I ensure that it is understandable by making appropriate nomenclature
- I review test cases regularly and update them if necessary.

NOTE: As can be seen from all these points, a test case that is clear, simple and has clear inputs and outputs would be close to ideal.

35. What are the Differences Between Build and Release?

A build is an executable file submitted by developers to the test team for testing the program. The program goes through several stages of repair and testing until it works properly. Once the program is stable and ready for end users, it is released.

A release is an installable program that is made available to end users after it has been approved by the test team. When any program is released to the customer, release notes are attached, describing the number of defects still outstanding, user stories covered, change needs and the release version.

36. What is Test Data and Why is it Important?

Test data is the input data used for software testing. This data is used to test a specific functionality or scenario of the software. Test data is used to simulate the real-world conditions of a test case and allows us to observe how the software will behave in a real environment.

Test data is important because the correct input data must be used for the software to work correctly. Testing with the wrong data can give misleading results and misleading information about how the software will work under real-world conditions. For example, if we use only small numbers as test data in software testing a calculation, we will not learn how the software will handle large numbers and we may get unexpected results when working with large numbers.

Let's give an example test data for UI tests: For example, if you want to log in to the system, you should be given a valid username and password. These are valid test data that you will use to log in to the system.

37. What is the difference between quality control and quality assurance?

Quality assurance (QA) and quality control (QC) are two important components of software testing that differ in their goals, focus and methodologies.

Quality assurance is a proactive technique that ensures that the software development process adheres to defined standards and processes and that the end product meets the required quality requirements. QA aims to prevent defects by discovering and addressing issues throughout the development process and verifying that the software meets customer needs and expectations. This includes actions such as determining test strategies and implementing test procedures to guarantee that the software is produced in accordance with the intended standards.

In contrast, quality control is a reactive way of ensuring that the software product meets the required quality requirements. Quality control involves executing test cases and

evaluating the results to find defects in the program. Functional, performance, security and other types of testing are performed to guarantee that the program meets quality standards. Quality control also includes measuring and evaluating software metrics to discover areas for improvement.

To summarize, quality assurance is about ensuring that the development process produces high quality software, while quality control is about testing and reviewing the software output to guarantee that it meets quality requirements. QA is proactive, while QC is reactive.
NOTE: Proactive means anticipating and taking precautions. This style of behavior means anticipating problems that may arise in the future and taking measures accordingly. Reactive means intervening after problems have occurred. This approach is used in emergency situations where a quick response is required.
In summary, a proactive approach means anticipating and taking precautions, while a reactive approach means intervening after problems have occurred.

38. When is Test Execution Stopped?

I should stop test execution when I realize that a serious problem, an important piece of functionality or an important system component is completely broken. For example, if I can't test other modules because of a bug in one module, it stops test execution. Such critical bugs are bugs that have a negative impact on the application.

1. The "Sign In" button in the Gmail App is not working and Gmail users cannot access their accounts.

2. When a consumer clicks on the money transfer button on a banking website, an error message is displayed.

39. What is the Difference Between Smoke and Sanity Testing?

Smoke testing is done to determine if the build obtained from the development team is testable. It is also known as a "Day 0" check. It is performed at the "build level". It helps avoid wasted testing time by only evaluating the entire program when critical functions are not working or important defects have not yet been resolved.

Sanity testing is performed during the release process to verify the core functionality of the application without delving too deeply into it. It is sometimes called a subset of regression testing. It is completed at the *"release"* level.

Let's explain the difference between the two tests with an example scenario through the E-commerce project:

Sample Test Scenario: Loading the home page (Smoke)

Step 1: The application home page opens.

Step 2: Check whether the page loaded as fast as expected.

Step 3: If the page load was successful, the main functions on the page (product categories, search box, favorites list) are checked.

Step 4: Verify that all main functions are working.

Sample Test Scenario: Product Search (Sanity)

Step 1: The application home page opens.

Step 2: Enter a product name in the search box.

Step 3: The search results page opens.

Step 4: Check whether the products shown on the page are filtered according to the search query.

Step 5: If the desired product is found, proceed to the product page and verify that the specifications on the page are correct.

40. What is Ad-Hoc Testing?

Ad-hoc testing is diametrically opposed to formal testing. It is a type of informal testing. Ad hoc testing involves testers randomly testing the program without any documentation or test design methodology. This testing is usually performed when testers have a thorough understanding of the program under test. Testers randomly test the program without any test case or business requirement documentation.

41. What are Integration Testing Approaches?

Integration testing is a critical step in software testing that evaluates the interaction of various modules, components and subsystems of a software system. There are numerous methods for performing integration testing, such as

Big Bang Integration Testing: This method requires all components to be tested together after they have been produced. With this strategy, the test team waits until all components are ready before testing. This method can save time during the testing process, but it can be difficult to detect and fix bugs.

Top-down Integrated Testing: In this method, the top-level modules are tested first, followed by the lower-level modules that support them. This method requires the use of stubs or drivers to emulate the behavior of lower-level modules. This can be useful.

Bottom-up Integration Testing: This method is the opposite of top-down integration testing; testing starts with lower-level modules and progresses to higher-level modules. In this method, actual modules are used instead of sketches or drivers, which can help detect bugs early. With this strategy, small pieces of code or modules are tested at a time and then gradually integrated into the system. This strategy allows the test team to find defects early and fix them before assembling the whole system.

Sandwich Integration Testing: This test method mixes top-down and bottom-up integration. First the top-level modules are tested, then the lower-level modules and finally the top-level modules again. This method is time consuming, but provides comprehensive results.

NOTE: Whichever of these approaches is suitable for the project is selected and applied.

42. What global and local testing means in software testing.

Global testing is testing to ensure that software can be properly adapted for different languages, cultures and regions without technical or functional problems. Global testing is also referred to as **internationalization** testing. The purpose of global testing is to ensure that the software product can support multiple languages and cultural traditions (such as date and time formats, number formats and currency symbols).

Local testing is testing to ensure that the software product is adapted for a specific region. The purpose of local testing is to ensure that the software product is culturally appropriate and relevant for the target market and that it works correctly with the specific language, formatting and other conventions used in that region.

In a nutshell, global testing focuses on ensuring that the software product can be easily adapted for different cultures and regions around the world, while local testing focuses on making sure that the software product is fully functional and culturally appropriate for a specific target market.

43. How much testing is needed?

There is no definitive answer to this question. In this direction, we cannot say that there is an absolute limit to the number of tests. Here, it would be the best practice to determine the most likely situations that can cause the greatest damage to the project-oriented application or the most frequently used areas of the program. We can use **risk-based testing** here. So we can focus our time and energy on the most critical parts. Testing should also provide enough information about the state or health of an application for stakeholders to make an educated choice about whether to release the program or spend additional time testing.

44. What would your attitude be if requirements changed frequently?

The purpose of the requirements does not change, but some technical details may change in the implementation. To deal with this situation, it is first necessary to focus on the purpose of the application.

First, it is necessary to determine how much requirements have changed. In this context it will be necessary to determine the extent and significance of the changes.

It is important to stay in regular contact with customers or users and focus on their needs. Also, by collaborating with project stakeholders, you can manage new requirements in the best possible way.

Prioritize changing needs and plan work according to these priorities. This ensures that the most important needs of customers or users are met.

Testing should be done frequently to make sure that the changes are working correctly. This way it can be checked whether the changes have any undesirable effects.

Respond flexibly to changing requirements. When the project needs to change direction or add new features, build a team that can adapt to these changes quickly and effectively.

As a result, it is important to adopt a flexible and open-minded approach to deal with changing requirements. It should be recognized that the needs of customers or users may change and that it is necessary to respond quickly to these changes.

45. What is Requirement Traceability and Why is it Important?

Requirement Traceability is a process in software testing that involves associating requirements with tests designed to verify that they are met. The goal of requirement traceability is to ensure that all requirements are tested and that any defects found during testing can be traced back to the original requirement.

The steps for requirements traceability can be listed as follows:

Defining requirements: The first step in requirements traceability is to identify all requirements for the software system under test.

Create a traceability matrix: A traceability matrix is a document that links requirements to test cases designed to verify them. Each requirement is assigned a unique identifier and each test case is linked to the requirement(s) it validates.

Execution of test cases: Once test cases have been identified and linked to requirements, they can be executed to verify that the software system meets the specified requirements.

Monitoring bugs: During testing, bugs may be discovered that need to be fixed. By tracing these bugs back to the original requirement, the development team can ensure that the requirement is met once the bug is fixed.

Updating the traceability matrix: As testing progresses, the traceability matrix should be updated to reflect changes or updates to requirements or test cases.

Requirements traceability helps to ensure that all requirements are tested, which can improve the overall quality of the software system. It also provides a way to trace defects back to their source, which can help identify patterns and improve the development process for future projects.

46. What Should be in a Requirement Traceability Example?

A requirements tracking chart, also known as a traceability matrix, is a tool used in software testing to track the relationship between requirements and test cases.

A Requirement Traceability Example should have Requirement ID, Description, Test Case ID, Test Case Description and Test Result as shown in the table below.

Requirement ID	Requirement Description	Test Case ID	Test Case Description	Test Result
REQ-001	The system must allow users to log in	TC-001	Verify that the login page is displayed	Pass
REQ-001	The system must allow users to log in	TC-002	Verify that the user can enter their username and password	Pass
REQ-001	The system must allow users to log in	TC-003	Verify that an error message is displayed when the user enters an incorrect password	Fail
REQ-002	The system should allow users to search for products	TC-004	Verify that the search page is displayed	Pass
REQ-002	The system should allow users to search for products	TC-005	Verify that the user can enter a search term	Pass
REQ-002	The system should allow users to search for products	TC-006	Verify that search results are displayed	Pass

In this example, the first column lists the requirement ID, the second column lists the requirement, the third column lists the test case ID, the fourth column lists the test case and the fifth column lists the result of the test case.

Each requirement is listed once and linked to all test cases designed to validate it. The graph shows which test cases have been executed and the results of each test case. If a test case fails, it is easy to identify which requirement it is tied to, allowing the development team to

quickly address the issue. The chart can be updated throughout the testing process to reflect changes or updates to requirements or test cases.