Manual Testing and Scenario based Interview Questions for Interview Selection Process

Q1: What is software testing?

- **It is a process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.**
- **It is a verification and validation process.**
- **Process of demonstrating that errors are not present.**

Q2: What is the difference between verification and validation?

Verification:

- **It is a process of confirming whether the software meets its requirement or not.**
- **Process of examining/reviewing work product.**
- **It is a QA activity.**
- **It's a static process performed at compile time.**
- **It is performed by a QA team or by a developer.**
- **Cost and time effective.**
- **Activities involved in this are testing the application.**

Validation:

- **It is a process of confirming whether the s/w meets the user's requirement or not.**
- **Process of executing a product & examining how it behaves.**
- **Are we building the right product?**
- **It is a QC activity.**
- **It's a dynamic process performed at run time.**
- **It is performed by a QC team or by tester.**
- **Cost and time taking.**
- **Activities involved in this are inspections, reviews, and walk-throughs.**

Q3: What is the difference between quality assurance and quality control?

Quality Assurance:

- It ensures the prevention of defects in the process used to make software applications.
- It involves process-oriented activities.
- Aim to prevent defects.

Eg:- verification

- It is the technique of managing the quality.
- All team members are responsible for QA.
- QA is responsible for SDLC.
- It is a process to create the deliverables.

Quality Control:

- It executes the program or code to identify the defects in the software application.
- It involves product-oriented activities.
- Aim to identify and improve.

Eg:- validation

- It is a method to verify the quality.
- The testing team is responsible for QC.
- QC is responsible for STLC.
- It is a process to verify the deliverables.

Q4: What is SDLC?

The Software Development Life Cycle refers to all the activities that are performed during software development, including requirement analysis, design, implementation, testing, deployment, and maintenance phases.

Q5: Explain STLC – Software Testing Life Cycle.

The software testing life cycle refers to all the activities performed during the testing of a software product. The phases include –

- **Requirement analyses and validation**: In this phase, the requirements documents are analyzed and validated, and the scope of testing is defined.

- **Test planning**: In this phase, the test plan strategy is defined, the estimation of test effort is defined along with the automation strategy, and tool selection is done.
- **Test Design and Analysis:** In this phase, test cases are designed, test data is prepared, and automation scripts are implemented.
- **Test environment setup**: A test environment closely simulating the real-world environment is prepared.
- **Test execution**: The test cases are prepared, and the bugs are reported and retested once resolved.
- **Test closure and reporting**: A test closure report is prepared with the final test results summary, learning, and test metrics.

Q6: What is dynamic testing?

It involves in the execution of code and validates the output with the expected outcome.

Q7: What is static testing?

It involves in reviewing the documents to identify the defects in the early stages of SDLC.

Q8: What is white box testing?

- This is also called glass-box testing, clear-box, and structural testing.
- It is based on the applications' internal code structure.
- In this, an internal perspective of the system, as well as programming skills are used to design test cases.
- In white box testing, the tester analyses the internal architecture of the system as well as the quality of source code on different parameters like code optimization, code coverage, code reusability, etc.
- This testing usually was done at the unit level.

Q9: What is black box testing?

- It is a process of testing a system component considering input, output, and general function.
- The tester interacts with the system through the interface providing input and validating the received output.
- It doesn't require knowledge of internal program structure.
- In this, we test UI & backend (coding/database).

- **External actions are performed.**

Q10: What is positive and negative testing?

Positive:

- **It determines what the system is supposed to do.**
- **It helps to check whether the application is justifying the requirements or not.**

Negative:

- **It determines what the system is not supposed to do.**
- **It helps to find the defects in the software.**

Q11: What is gray box testing?

It is a combination of both the black-box and white-box testing. The tester who works on this type of testing needs to have access to design documents, this helps to create better test cases.

Q12: What is a test strategy?

It is a high-level document and is usually developed by the project manager. It's a document that captures the approach about how we go about testing the product and achieving the goals.

Q13: What is test plan?

It is a document which contains the plan for all the testing activities.

Q14: What is test scenario?

It gives the idea of what we have to test or the testable part of an application is called TS.

Q15: What is test case?

It is a set of conditions under which tester determines whether an application/ software is working correctly or not.

Q16: What is test bed?

An environment configured for testing is called test bed. It consists of hardware, s/w, network configuration.

Q17: What is test suite?

A collection of test cases is called as test suite.

Q18: What is test data?

It is a document that is basically used to test the s/w program. It is divided into 2 categories: -

1. **+ve test data which is generally given to the system to generate the expected result.**
2. **–ve test data which is used to test**

Q19: What is defect life cycle?

Defect Life Cycle or Bug Life Cycle is the specific set of states that a Bug goes through from discovery to defect fixation.

Bug Life Cycle phases/status: - The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

- **New: When a new defect is logged and posted for the first time. It is assigned a status as NEW.**
- **Assigned: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team.**
- **Open: The developer starts analyzing and works on the defect fix.**
- **Fixed: When a developer makes a necessary code change and verifies the change, he or she can make the bug status "Fixed."**
- **Pending retest: after fixing the defect the developer gives a particular code for retesting the code to the tester. Here the testing is pending on the tester's end, the status assigned is "pending request."**
- **Retest: Tester does the retesting of the code, to check whether the defect is fixed by the developer or not and changes the status to "Re-test."**
- **Verified: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."**

- **Reopen: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.**
- **Closed: If the bug no longer exists then the tester assigns the status "Closed."**
- **Duplicate: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to "duplicate."**
- **Rejected: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."**
- **Deferred: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then the status "Deferred" is assigned to such bugs**
- **Not a bug: If it does not affect the functionality of the application then the status assign to a bug is "Not a bug".**

- **New – A bug or defect when detected is in a new state**
- **Assigned – The newly detected bug when assigned to the corresponding developer is in the Assigned state**
- **Open – When the developer works on the bug, the bug lies in the Open state**
- **Rejected/Not a bug – A bug lies in rejected state in case the developer feels the bug is not genuine**
- **Deferred – A deferred bug is one, fix which is deferred for some time (for the next releases) based on the urgency and criticality of the bug**
- **Fixed – When a bug is resolved by the developer it is marked as fixed**
- **Test – When fixed the bug is assigned to the tester and during this time the bug is marked as in Test**
- **Reopened – If the tester is not satisfied with the issue resolution the bug is moved to the Reopened state**
- **Verified – After the Test phase if the tester feels the bug is resolved, it is marked as verified**
- **Closed – After the bug is verified, it is moved to Closed status.**

Q20: What is smoke and sanity testing?

Smoke:

- **It is a kind of Software Testing performed after the software built to ascertain that the critical functionalities of the program are working fine.**

- **The purpose is to reject a badly broken application so that the QA team does not waste time installing and testing the software application.**
- **In Smoke Testing, the test cases chose to cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.**

Sanity Testing:

- **Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.**

Q21: What is an exit and entry criteria?

Entry:

It describes when to start testing i.e. what we have to test it should be stable enough to test.

Ex:- if we want to test the home page, the SRS/BRS/FRS document & the test cases must be ready and should be stable enough to test.

Exit:

It describes when to stop testing i.e. once everything mentioned below is fulfilled then s/w release is known as exit criteria:-

- **Followed before actually releasing the software to the client. Checking whether the computer testing is done or not.**
- **Documents checking:- test matrix (RTM)/summary reports.**

SUSPENSION CRITERIA→ when to stop testing temporarily.

Q22: What is a blocker?

A blocker is a bug of high priority and high severity. It prevents or blocks testing of some other major portion of the application as well.

Q23: What is regression testing?

To test whether the changed component has introduced any error to unchanged component or not is called as regression testing. It is perform on QA/production site depends.

Q24: What is retesting?

To test whether the reported bug has been resolved by the developer team or not, is known as retesting.

Q25: What is monkey / ad-hoc testing?

It is an informal testing performed without a planning or documentation and without having knowledge of the applications/software functionalities. Monkey testing is a type of testing that is performed randomly without any predefined test cases or test inputs.

Q26: What is the difference between severity and priority?

Priority:

- **“How prior we need to fix the bug is priority.”**
- **It means the occurrences of defect.**
- **Decide by developer team. Types (low, medium, high, critical)**

Severity:

- **“How severe the bug is severity”.**
- **It means how bad the defect is and what impact it can cause in our application.**
- **Decide by the testing team. Types (minor, medium, major)**

Q27: What is defect priority?

A defect priority is the urgency of the fixing the defect. Normally the defect priority is set on a scale of P0 to P3 with P0 defect having the most urgency to fix.

Q28: What is defect severity?

Defect severity is the severity of the defect impacting the functionality. Based on the organization, we can have different levels of defect severity ranging from minor to critical or show stopper.

Q29: Give an example of Low Priority-Low severity, Low Priority-High severity, High Priority-Low severity, High Priority-High severity defects.

1. Low Priority-Low severity – A spelling mistake in a page not frequently navigated by users.

2. Low Priority-High severity – Application crashing in some very corner case.

3. High Priority-Low severity – Slight change in logo color or spelling mistake in company name.

4. High Priority-High severity – Issue with login functionality.

Q30: What is unit testing?

- **It is also called as module testing /component testing.**
- **It is done to check whether the individual unit or module of the source code is working properly.**
- **It is done by the developer.**

Q31: What is integration testing?

- **It is a process of testing the interface between the two s/w units.**
- **It is done by 3 ways: - big-bang, top-down, bottom-up approach.**
- **Process of combining & testing multiple components together.**
- **Normally done by developer but a tester can also perform if he has the knowledge of coding.**

Q32: What is system testing?

- **It is a black box testing technique performed to evaluate the computer system. It includes both functional and non-functional testing. Verifying the completed system to ensure that the application works as intended or not.**
- **“The behavior of the system is tested as defined by the scope of the development project”.**

- Carried out by specialist tester/independent tester.

Q33: What is user-appearance testing?

- User-requirement testing is done.
- Done by client as well as end user.
- It is a final stage of testing before used.

Q34: What is alpha-beta testing?

Alpha Testing:

- Developer records all the issues.
- Done by the end user at dev site. (involves client or tester + dev)

Beta Testing:

- Dev go through all the issues after specific period of time.
- Done by the end user at the client site. (involves client/user)

Q35: How monkey testing is different from adhoc testing?

In case of adhoc testing although there are no predefined or documented test cases still testers have the understanding of the application. While in case of monkey testing testers doesn't have any understanding of the application.

Q36: Explain TDD (Test Driven Development).

Test Driven Development is a software development methodology in which the development of the software is driven by test cases created for the functionality to be implemented. In TDD, first the test cases are created and then code to pass the tests is written. Later the code is re-factored as per the standards.

Q37: Explain Equivalence Class Partitioning.

Equivalence class partitioning is a specification based black box testing techniques. ECP means Grouping test data into equivalence classes with the assumption that all the data items lying in the classes will have same effect on the application. In simple it means diving any module into equal parts and test each part separately.

Example:

1. For testing a Square – Program (program that prints the square of a number- the equivalence classes can be: -Set of Negative numbers, whole numbers, decimal numbers, set of large numbers etc.)
2. Suppose we have to test 1-100 no's. So, 1st we will divide this no into 5 equal parts. (Like 1-20, 21-40,41-60,61-80,81-100). Now we will select random 3 values and multiply these values with the no of parts. Whatever the no will be, we will check for those values from all the module in place of checking 100 values.

Purpose: testing a complete module is exhaustive testing and time-consuming that's why we use equivalence partitioning, as it is time saving.

Q38: What is boundary value analysis?

Boundary value analysis is a software testing technique for designing test cases wherein the boundary values of the classes of the equivalence class partitioning are taken as input to the test cases. It is also called as a part of stress and –ve testing.

e.g. if the test data lies in the range of 0-100, the boundary value analysis will include test data – 0,1, 99, 100.

Q39: What are some defect reporting attributes?

Some of the attributes of a Defect report are

- DefectId – A unique identifier of the defect.
- Defect Summary – A one line summary of the defect, more like a defect title.
- Defect Description – A detailed description of the defect.
- Steps to reproduce – The steps to reproduce the defect.
- Expected Result – The expected behaviour from which the application is deviating because of the defect.
- Actual Result- The current erroneous state of the application w.r.t. the defect.
- Defect Severity – Based on the criticality of the defect, this field can be set to minor, medium, major or show stopper.
- Priority – Based on the urgency of the defect, this field can be set on a scale of P0 to P3.

Q40: What is a stub?

In case of top-down integration testing, many a times lower level modules are not developed while beginning testing/integration with top level modules. In those cases Stubs or dummy modules are used that simulate the working of modules by providing hard-coded or expected output based on the input values.

Q41: What is a driver?

In case of bottom-up integration testing, drivers are used to simulate the working of top level modules in order to test the related modules lower in the hierarchy.

Q42: What are some advantages of automation testing?

- **Some advantages of automation testing are Test execution using automation is fast and saves considerable amount of time.**
- **Carefully written test scripts remove the chance of human error during testing.**
- **Tests execution can be scheduled for nightly run using CI tools like Jenkins which can also be configured to provide daily test results to relevant stakeholders.**
- **Automation testing is very less resource intensive. Once the tests are automated, test execution requires almost no time of QAs. Saving Qa bandwidth for other exploratory tasks.**

Q43: What are some advantages of automation testing?

Some disadvantages of automation testing are

- **It requires skilled automation testing experts to write test scripts.**
- **Additional effort to write scripts is required upfront.**
- **Automation scripts are limited to verification of the tests that are coded. These tests may miss some error that is very glaring and easily identifiable to human(manual QA).**
- **Even with some minor change in application, script updation and maintenance is required Verification strategies/error guessing/sdlc/white box techniques/exhaustive testing/v model/spiral model/non-functional testing.**

Q44: Difference between Waterfall Model and Agile Methodology.

The Waterfall Model and Agile Methodology are two different approaches to software development:

Waterfall Model:

Sequential and linear approach
Fixed requirements at the beginning
Emphasis on documentation
Limited customer involvement
Late bug detection

Agile Methodology:

Iterative and incremental approach
Flexible and adaptive to changing requirements
Emphasis on collaboration and communication
Continuous testing throughout the process
Active customer involvement

The Waterfall Model is suitable for projects with stable requirements and a focus on predictability. Agile Methodology is ideal for projects requiring flexibility, collaboration, and the ability to adapt to evolving requirements.

Q45: What is a test plan, and what are the steps to create one?

A test plan is a document that outlines the strategy, scope, resources, and schedule for testing a product. It is an important part of the software development process, as it helps ensure that the product is of high quality and meets the requirements and specifications.

To create a test plan, you can follow these steps:

1. **Identify the goals of the testing. What do you want to achieve with the testing? What are the objectives of the test plan?**
2. **Define the scope of the testing. What features and functions of the product will be tested? What environments and platforms will the testing be conducted on?**
3. **Determine the resources needed for testing. What personnel, equipment, and tools will be required?**

4. **Develop a testing schedule**. When will the testing take place? How long will it take?
5. **Determine the test approach.** How will the testing be conducted? What types of testing will be used (e.g., unit testing, integration testing, system testing, acceptance testing)?
6. **Create a test matrix.** This is a table that maps the test cases to the requirements or functions being tested.

Integration Testing	System Testing
It enables you to assess, validate, and test the application design in addition to the business requirements.	**A form of testing known as "system testing" determines whether the system as a whole satisfies both functional and non-functional criteria.**
Integrity testing's primary goal is to find flaws in how components or sections communicate with one another.	**The primary goal of system testing is to confirm that the system satisfies the requirements and is suitable for use in the environment intended.**
Integration testing will primarily concentrate on the interfaces between components or modules.	**The general conduct of the system is examined during system testing.**
To make sure they function as intended, integration testing examines the relationships between components or subsystems.	**System testing includes putting the system through realistic conditions to make sure it performs as anticipated in the target setting.**
Typically, developers or testers who work closely with developers conduct integration testing.	**Typically, testers who are not members of the programming team perform system testing.**
White-box testing is a form of integration testing, which means that testers are familiar with the inner workings of the components or modules under test.	**System testing is a type of "black-box" testing, which indicates that the testers are blind to how the system actually works.**
Integration testing involves testing specific scenarios involving the interactions between components or subsystems.	**System testing involves testing the system in a range of scenarios to ensure it works as intended under different conditions.**

Integration testing is usually automated using testing frameworks and tools.	System testing is usually carried out manually, although some aspects may be automated using testing tools.
Integration testing is generally less expensive and less time-consuming than system testing.	System testing is generally more expensive and more time-consuming than integration testing.

7. **Write the test cases.** A test case is a set of steps and expected results that a tester follows to verify that a feature or function of the product is working correctly.
8. **Review and revise the test plan.** Make sure that the test plan is complete, accurate, and feasible.
9. **Execute the testing.** Follow the test plan and test cases to test the product.
10. **Document the results of the testing.** This includes any issues or defects that were found, and how they were addressed.

By following these steps, you can create a comprehensive and effective test plan that will help ensure the quality and reliability of your product.

Q46: What is the difference between system and integration testing?

Q47: What does verification mean?

S.NO.	Boundary value analysis	Equivalence partitioning
1.	It is a technique where we identify the errors at the boundaries of input data to discover those errors in the input center.	It is a technique where the input data is divided into partitions of valid and invalid values.
2.	Boundary values are those that contain the upper and lower limit of a variable.	In this, the inputs to the software or the application are separated into groups expected to show similar behavior.
3.	Boundary value analysis is testing the boundaries between partitions.	It allows us to divide a set of test conditions into a partition that should be considered the same.

4.	**It will help decrease testing time due to a lesser number of test cases from infinite to finite.**	**The Equivalence partitioning will reduce the number of test cases to a finite list of testable test cases covering maximum possibilities.**
5.	**The Boundary Value Analysis is often called a part of the Stress and Negative Testing.**	**The Equivalence partitioning can be suitable for all the software testing levels such as unit, integration, system.**
6.	**Sometimes the boundary value analysis is also known as Range Checking.**	**Equivalence partitioning is also known as Equivalence class partitioning.**

Verification includes different activities such as business requirements, system requirements, design review, and code walk-throughs while developing a product.

It is also known as static testing, where we are ensuring that "we are developing the right product or not". And it also checks that the developed application fulfilling all the requirements given by the client.

Q48: Difference between boundary value analysis and equivalence partitioning?

Q49: What is the difference between authorization and authentication?

Authentication	**Authorization**
Authentication is the process of identifying a user to provide access to a system.	**Authorization is the process of giving permission to access the resources.**
In this, the user or client and server are verified.	**In this, it is verified that if the user is allowed through the defined policies and rules.**
It is usually performed before the authorization.	**It is usually done once the user is successfully authenticated.**
It requires the login details of the user, such as user name & password, etc.	**It requires the user's privilege or security level.**
Data is provided through the Token Ids.	**Data is provided through the access tokens.**

Example: Entering Login details is necessary for the employees to authenticate themselves to access the organizational emails or software.	**Example: After employees successfully authenticate themselves, they can access and work on certain functions only as per their roles and profiles.**
Authentication credentials can be partially changed by the user as per the requirement.	**Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it.**

Q50: Explain the process of test case review.

The process of test case review involves the following steps:

- **Preparation: Test cases are prepared based on the requirements and specifications of the software. They are documented in a standardized format.**
- **Selection of Reviewers: A group of qualified reviewers is selected, including test leads, domain experts, developers, and other stakeholders with relevant knowledge and expertise.**
- **Review Meeting: A review meeting is conducted, either in person or virtually, where the reviewers gather to discuss and examine the test cases. A designated moderator or the test case author leads the meeting.**
- **Test Case Examination: Reviewers systematically analyze each test case, ensuring accuracy, coverage of scenarios, and alignment with requirements. They look for clarity, consistency, and adherence to best practices.**
- **Feedback and Discussion: Reviewers provide feedback, raise concerns, and ask questions during the meeting. The discussion clarifies doubts and establishes a shared understanding.**
- **Issue Identification and Resolution: Identified issues, defects, or improvements are documented. The responsible person addresses these issues, either during the meeting or afterward, to improve the quality of the test cases.**

- **Follow-up Actions: The test case author incorporates suggested changes, updates the test cases, and shares them with reviewers for final review or approval.**

The test case review process ensures the reliability and effectiveness of test cases, improving the overall quality of the testing effort.

Q51: Differentiate between regression and retesting.

Regression Testing	**Re-testing**
Regression Testing is carried out to confirm whether a recent program or code change has not adversely affected existing features	**Re-testing is carried out to confirm the test cases that failed in the final execution are passing after the defects are fixed**
The purpose of Regression Testing is that new code changes should not have any side effects to existing functionalities	**Re-testing is done on the basis of the Defect fixes**
Defect verification is not the part of Regression Testing	**Defect verification is the part of re-testing**
Based on the project and availability of resources, Regression Testing can be carried out parallel with Re-testing	**Priority of re-testing is higher than regression testing, so it is carried out before regression testing**
You can do automation for regression testing, Manual Testing could be expensive and time-consuming	**You cannot automate the test cases for Retesting**
Regression testing is known as a generic testing	**Re-testing is a planned testing**
Regression testing is done for passed test cases	**Retesting is done only for failed test cases**
Regression testing checks for unexpected side-effects	**Re-testing makes sure that the original fault has been corrected**
Regression testing is only done when there is any modification or changes become mandatory in an existing project	**Re-testing executes a defect with the same data and the same environment with different inputs with a new build**

Test cases for regression testing can be obtained from the functional specification, user tutorials and manuals, and defect reports in regards to corrected problems	**Test cases for retesting cannot be obtained before start testing.**

Q52: What is the purpose of ECP?

Testing a complete module is exhaustive testing and time consuming that's why we use Equivalence partitioning as it is time saving.

Q53: What are the important Black Box Test Design Techniques?

Black-box test design techniques:

- **Equivalence partitioning**
- **Boundary value analysis**
- **Decision Table Testing**
- **State Transition Testing**
- **Use Case Testing**

Q54: What is random testing?

It is a Block box test design technique and informal one.

Q55: What is the purpose of Test Design Technique?

The purpose of test design techniques is to identify test conditions and test scenarios through which effective and efficient test cases can be written. Using test design techniques is a best approach rather the test cases picking out of the air. Test design techniques help in
• Achieving high test coverage.
• Defining tests that will provide insight into the quality of the test object.

Q56: What is use case testing?

A black box test design technique in which test cases are designed to execute User scenarios of Business scenarios.

Q57: What is the equivalence test design technique?

It is a Black-box (Specification Based) Test Case Design Technique with two primary goals-
1. To reduce the number of test cases to the necessary minimum.
2. To select the right test cases to cover all possible scenarios.

Q58: What is State Transition Testing?

A system may exhibit a different response depending on current conditions or previous history (its state). In this case, that aspect of the system can be shown with a state transition diagram. It allows the tester to view the software in terms of its states, transitions between states, the inputs or events that trigger state changes (transitions), and the actions that may result from those transitions. The states of the system or object under test are separate, identifiable, and finite in number.

Q59: What are the different types of test design techniques?

Different types of test design techniques:

Test design techniques are categorized into two types. They are:
1. Static testing technique.
2. Dynamic testing technique.

The dynamic techniques are subdivided into three more categories. They are:
1. Specification-based (black-box, also known as behavioral techniques) techniques.
2. Structure-based (white-box or structural techniques) techniques.
3. Experience-based techniques.

The Specification-based or Black-box testing techniques are
a. Equivalence partitioning.
b. Boundary value analysis.
c. Decision tables.
d. State transition testing.
e. Use case testing.

Structure-based or White-box testing techniques are Statement Testing and Coverage, Decision Testing and Coverage, and Linear Code Sequence And Jump (LCSAJ).

Q60: What is Walkthrough in static technique?

Walk-through:

• It is not a formal process/review
• It is led by the authors
• Author guides the participants through the document according to his or her thought process to achieve a common understanding and to gather feedback.
• Useful for the people if they are not from the software discipline, who are not used to or cannot easily understand software development process.
• Is especially useful for higher level documents like requirement specification, etc.

The goals of a walkthrough:
i. To present the documents both within and outside the software discipline in order to gather the information regarding the topic under documentation.
ii. To explain or do the knowledge transfer and evaluate the contents of the document
iii. To achieve a common understanding and to gather feedback.
iv. To examine and discuss the validity of the proposed solutions

Q61: What is a Technical Review in static technique?

The aim of this review technique is to achieve consensus about the technical aspect of the document. They are informal in nature and it is the experts, who identify defects in the document. The experts who are a part of the review are architects, chief designers, key users, etc. However, peers are also a part of the review as well. In a technical review, the value of the technical concepts and alternatives is assessed. It is also ensured that the right technical concepts are used.

Q62: What is a Static testing technique?

Static test techniques provide a great way to improve the quality and productivity of software development. It includes the reviews and provides the overview of how they are conducted. The primary objective of static testing is to improve the quality of software products by assisting engineers to recognize and fix their own defects early in the software development process.

Q63: What are the uses of Static Testing?

The uses of static testing are as follows:

- **Since static testing can start early in the life cycle so early feedback on quality issues can be established. As the defects are getting detected at an early stage so the rework cost most often relatively low. Development productivity is likely to increase because of the less rework effort.**
- **Types of the defects that are easier to find during the static testing are: deviation from standards, missing requirements, design defects, non-maintainable code and inconsistent interface specifications.**
- **Static tests contribute to the increased awareness of quality issues.**

Q64: Can u tell me about the kick-off in a formal review?

This kick-off meeting is an optional step in a review procedure. The goal of this step is to give a short introduction on the objectives of the review and the documents to everyone in the meeting. The relationships between the document under review and the other documents are also explained, especially if the numbers of related documents are high. At customer sites, we have measured results up to 70% more major defects found per page as a result of performing a kick-off.

Q65: What is a formal review?

Formal reviews follow a formal process. It is well structured and regulated. A formal review process consists of six main steps:

1. Planning
2. Kick-off
3. Preparation
4. Review meeting
5. Rework
6. Follow-up

Q66: What is informal review in static technique?

Informal reviews are applied many times during the early stages of the life cycle of the document. A two-person team can conduct an informal review. In later stages these reviews often involve more people and a meeting. The goal is to keep the author and to improve the quality of the document. The most important thing to keep in mind about the informal reviews is that they are not documented.

Q67: What is specification-based technique?

Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.

Q68: What is a decision table testing?

A black box test design technique in which test cases are designed to execute the combinations of inputs and/or stimuli (causes) shown in a decision table.

Q69: What are the advantages of decision table testing?

Advantages of decision table testing:
1) A decision table provides a framework for a complete and accurate statement of processing or decision logic.
2) Helps to identify the test scenarios faster because of tabular representation.
3) Easy to understand.
4) Easy to maintain and update decision table if there is change in requirement.
5) It is possible to check that all test combinations have been considered.

Q70: What is Experience-based technique?

Experience-based tests utilize testers’ skills and intuition, along with their experience with similar applications or technologies. These tests are effective at finding defects but are not as appropriate as other techniques to achieve specific test coverage levels or produce reusable test procedures.

Q71: What are the experience-based test design techniques?

Experience-based test design techniques:

- **Error guessing.**
- **Checklist-based.**
- **Exploratory.**

Q72: What is the boundary value analysis test design technique?

Boundary value analysis testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain. Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

Q73: What is Exploratory testing?

Exploratory testing is used in order to gain knowledge required for designing appropriate and effective tests. Exploratory testing produces test conditions with each iteration of the development lifecycle.

Q74: What is Equivalence partitioning?

Equivalence partitioning is a software testing technique that groups the input data for a module of a software into partitions of data that can represent each possible case. Then, select an input data from each partition.

Q75: What are the different levels of testing?

There are mainly four testing levels and they are:

- **Unit Testing**
- **Integration Testing**
- **System Testing**
- **Acceptance Testing**

Q76: What are the categories of defects?

There are three main categories of defects as shown in the below figure:

- **Wrong: It implies that requirements have been implemented incorrectly. It is a variance from the given specification.**
- **Missing: This is a variance from the specifications, an indication that a specification was not implemented, or a requirement of the customer was not noted properly.**
- **Extra: It is a requirement incorporated into the product that was not given by the end customer. It is always a variance from the specification but may be an attribute desired by the user of the product**

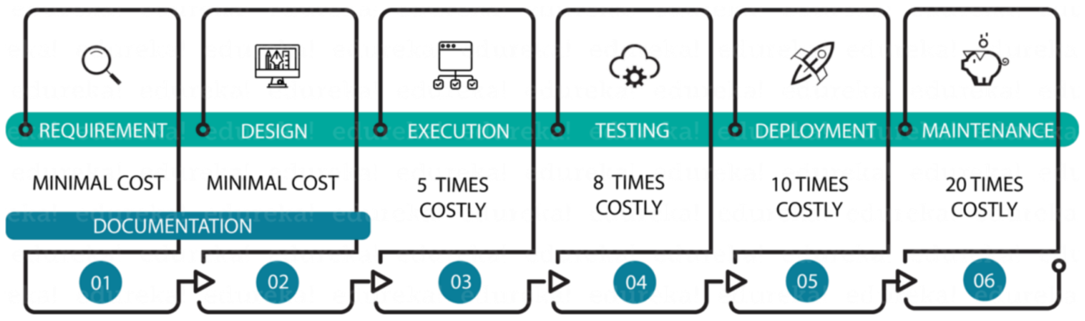
Q77: On what basis the acceptance plan is prepared?

Basically, the acceptance document is prepared using the following inputs.

- **Requirement document:** It specifies what exactly is needed in the project from the customer's perspective.
- **Input from the customer:** This can be discussions, informal talks, emails, etc.
- **Project plan:** The project plan prepared by the project manager also serves as good input to finalize your acceptance test.

Q78: A defect which could have been removed during the initial stage is removed in a later stage. How does this affect the cost?

If at the initial stage a defect is identified, then it should be removed during that stage/phase itself rather than at some later stage. It's a fact that if a defect is delayed for later phases, it becomes more costly. The following figure shows how a defect is costly as the phases move forward.



If a defect is identified and removed during the design phase, it is the most cost effective but when removed during maintenance it becomes twenty times costlier.

Q79: On what basis you can arrive at an estimation for your project?

To estimate your project, you have to consider the following points:

- Divide the whole project into the smallest tasks
- Allocate each task to team members
- Estimate the effort required to complete each task
- Validate the estimation

Q80: What is performance testing?

Performance testing is a discipline where we test our system to evaluate the system performance when subjected to virtual user load.

Q81: How is Load Testing different from Stress Testing?

Load Testing: It is the simplest form of testing which is done by increasing load step by step to reach the defined limit or goal.
Stress Testing: We can also call it as negative testing as we test the system beyond its capacity to find out the break point of the system.

Q82: What is concurrent user load in performance testing?

Concurrent user load can be defined as when multiple users hit to any functionality or transaction at the same time.

Q83: What can be key factors in deciding whether we need a performance testing for any application before going live?

We can decide based on the following factors:

- How many are the target users?
- How much application will grow with period of time?
- Who all are competitors and how much traffic they general get?
- How many critical scenarios we have?
- How much will be the business loss if site is unavailable?

Q84: What is the Purpose of Security Testing?

Security testing is a sort of software testing that identifies vulnerabilities, hazards, and dangers in a software program and guards against intruder assaults. The goal of security tests is to find any potential flaws and vulnerabilities in the software system that might lead to a loss of data, income, or reputation at the hands of workers or outsiders.

Examples of Security Testing Scenarios:

Sample test scenarios to give you an idea of the kind of security tests that are available −

- A password must be stored in an encrypted way.
- Invalid users should not be allowed to access the application or system.

- **For application, check cookies and session time.**
- **The browser back button should not operate on financial sites.**

Q85: What is soak or endurance testing?

Soak testing (otherwise known as endurance testing, capacity testing, or longevity testing) involves testing the system to detect performance-related issues such as stability and response time by requesting the designed load on a system.

Examples of Soak Testing:

Ideally, a soak test should run as long as possible, though it will depend on the system for you to determine the duration of your test. Below are some examples when soak testing's can be considered:

When a bank announces that it will be closing, its system is expected to handle a large number of transactions during the closing days of the bank. This event is rare and unexpected, but your system has to handle this unlikely situation anyway.

Q86: What is spike testing and why it is needed?

Spike testing is a type of performance testing in which an application receives a sudden and extreme increase or decrease in load.

The goal of spike testing is to determine the behavior of a software application when it receives extreme variations in traffic.

Q87: Difference Between Functional and Non-Functional Testing.

Functional Testing	Non-Functional Testing
It tests 'What' the product does. It checks the operations and actions of an application.	**It checks the behavior of an Application.**
Functional testing is done based on the business requirement.	**Non- functional testing is done based on the customer expectation and Performance requirement.**

Functional Testing	Non-Functional Testing
It tests whether the actual result is working according to the expected result.	**It checks the response time, and speed of the software under specific conditions.**
It is carried out manually. Example: Black box testing method.	**It is more feasible to test using automated tools. Example: LoadRunner.**
It tests as per the customer requirements.	**It tests as per customer expectations.**
Customer feedback helps in reducing the risk factors of the product.	**Customer feedback is more valuable for non-functional testing as it helps to improve and lets the tester to know the expectation of the customer.**
It is testing the functionality of the software.	**It is testing the performance of the functionality of the software.**
Functional testing has the following types: •Unit testing •Integration testing •System Testing •Acceptance Testing	**Nonfunctional testing includes: •Performance testing •Load Testing •Stress testing •Volume testing •Security testing •Installation testing •Recovery testing**
Example: A Login page must show textboxes to Enter the username and password.	**Example: Test if a Login page is getting loaded in 5 seconds.**

Q88: How do you map STLC to SDLC? Specify what testing activities are held in each phase of SDLC.

As previously mentioned, it is recommended that testing activity starts as early as possible in the corresponding SDLC phase. In real life, though, it does not always happen this way.

Phase of SDLC	Phase of STLC	Testing Activities

Planning	**Requirements Analysis**	**Analyze and test the requirementsEvaluate automation feasibility**
Analysis and Design	**Test Planning**	**Create test planEvaluate tools and resourcesEstimate test effort**
Implementation	**Test Design and Environment Setup**	**Create test cases and test scriptsPrepare test environmentSetup test dataPerform a smoke test**
Testing	**Test Execution**	**Execute test cases and test scriptsLog bugsPerform different types of testingDo regression testingVerify bug fixes**
Deployment and Maintenance	**Test Closure**	**Prepare test reportsProvide test metrics to stakeholdersHold Lessons Learned sessions and come up with action items for future cyclesArchive test artifacts and results**

Q89: Who prepares Test Cases?

In System Testing Level, Testers prepare test cases under the guidance of the Test Lead.

Q90: What is the relation between Test case and Test data?

We use Test data as input for executing test cases

Q91: How to collect Test Data?

Some test data we (testers) prepare, some Test data we collect from Developers, and some Test Data we collect from Customers and other sources.

Q92: Is test data mandatory for every test case?

Test Data is not mandatory for every test case.

Q93: How do we derive Test cases?

We derive test cases from Test scenarios, test scenarios derived from requirements.

Q94: How to write good test cases?

Writing good test cases:

- **Analyzing requirements – To write a good test case, a tester needs to understand the requirement. In what context the requirement is described and what needs to be tested and how. What must be the expected result etc?**
- **Writing test cases (test designing) – A test case is developed based on the high level scenarios, which are in turn developed from the requirement. So, every requirement must have at least one test case. This test case needs to be wholly concentrated on the requirement.**

For ex: Lets take yahoomail.com, in this website, the requirement says that username can accept alphanumeric characters. So, the test case must be written to check for different combinations like testing with only alphabets, only numeric and alphanumeric characters. And the test data that you give for each test case is different for each combination. Like this, we can write any number of test cases, but optimization of these test cases is important. Optimize exactly what all test cases we need and whatnot.

Q95: What is the difference between test scenario and test case?

- **A test case is a procedure followed to execute a test, it consists of the details about the input and expected result.**
- **Test scenarios are based on a hypothetical story to help a person to think through a complex program or a system.**

Test cases derived from Test Scenarios.

Q96: What is a positive and negative test case?

Positive test case is intended for Verifying a Component or System's behavior using Valid inputs. Negative test case is intended for Verifying a Component or System's behavior using Invalid inputs.

Q97: Explain Agile Methodology.

Agile methodology of software development is based on iterative and incremental approach. In this model, the application is broken down into smaller build on which different cross functional team work together

providing rapid delivery along with adapting to changing needs at the same time.

Working is done by individual person.

- **There is a scrum master, who will be either a tester/developer from the team or a person who has knowledge of both testing and coding.**
- **The responsibility of the scrum master is to narrate the stories to both the team i.e. testing team and the development team.**
- **Scrum meetings can happen once a week or in 15 days or once a month. Most of the time client is included in scrum meeting.**
- **Because of this meeting, if one person is absent another person from the same team can complete his work. So the project isn't paused and dependency on one person does not happen. This is the main advantage of this model.**
- **Sprint is dividing the project into modules and distributing these modules among both teams so that the team is working in parallel.**
- **When to use: when the project is big/medium and we have to deliver it as soon as possible then we will use this model. Quality is maintained.**

Q98: What is scrum?

A scrum is a process for implementing Agile methodology. In scrum, time is divided into sprints and on completion of sprints, a deliverable is shipped.

Q99: What are the different roles in scrum?

The different roles in scrum are –

- **Product Owner – The product owner owns the whole development of the product, assigns tasks to the team, and acts as an interface between the scrum team(development team) and the stakeholders.**
- **Scrum Master – The scrum master monitors that scrum rules get followed in the team and conducts scrum meetings.**
- **Scrum Team – A scrum team participate in the scrum meetings and perform the tasks assigned.**

Q100: What is a scrum meeting?

A scrum meeting is a daily meeting in the scrum process. This meeting is conducted by the scrum master and an update of the previous day's work along with the next day's task and context is defined in this meeting.

Q101: Provide 20 test scenarios for the login page.

1. **Valid credentials: Enter a valid username and password and ensure successful login.**
2. **Invalid username: Enter an invalid username and a valid password and verify that an appropriate error message is displayed.**
3. **Invalid password: Enter a valid username and an invalid password and verify that an appropriate error message is displayed.**
4. **Empty username field: Leave the username field empty and enter a valid password. Confirm that an error message prompts for the username.**
5. **Empty password field: Leave the password field empty and enter a valid username. Verify that an error message prompts for the password.**
6. **Case sensitivity: Test if the login is case sensitive by entering the correct username and password but with different cases. Ensure that it recognizes the difference.**
7. **Long username: Enter a username with more than the maximum allowed characters and ensure it is handled correctly.**
8. **Long password: Enter a password with more than the maximum allowed characters and verify that it is handled properly.**
9. **Copy-paste: Copy and paste the username and password from an external source to test if the login page supports this functionality.**
10. **SQL injection: Attempt to enter SQL injection characters as the username and password and validate that they are properly sanitized and rejected.**
11. **Cross-site scripting (XSS): Enter XSS script tags as the username and password and verify that they are not executed and are displayed as plain text.**
12. **Forgot password: Click on the "Forgot password" link and confirm that the user is redirected to the password recovery page.**
13. **Remember me: Log in with the "Remember me" checkbox selected and verify that the session remains active even after closing and reopening the browser.**
14. **Concurrent logins: Attempt to log in with the same account from multiple devices or browsers simultaneously and ensure that it handles the scenario correctly.**

15. **Account lockout: Enter incorrect credentials multiple times and verify that the account gets locked after a certain number of failed attempts.**
16. **Social media login: Test the functionality of logging in with social media accounts (e.g., Facebook, Google) and ensure that the integration works correctly.**
17. **Accessibility: Verify that the login page is accessible to users with disabilities by using screen readers or other assistive technologies.**
18. **Browser compatibility: Test the login page on different browsers (Chrome, Firefox, Safari, etc.) and ensure consistent behavior and appearance.**
19. **Performance under load: Simulate a high load on the login page by sending multiple login requests simultaneously and verify that it responds efficiently.**
20. **Session timeout: Log in and wait for the session timeout period to expire. Confirm that the user is automatically logged out and prompted to log in again.**

These test scenarios cover a range of scenarios to ensure that the login page functions properly and handles different situations gracefully.

Q102: Write 10 test cases for adding an item to the cart in an e-commerce application.

1. **Valid item: Select a valid item from the product catalog and verify that it is successfully added to the cart.**
2. **Empty cart: Attempt to add an item to an empty cart and ensure that the cart is updated with the added item.**
3. **Multiple items: Add multiple items to the cart and confirm that all the selected items are accurately displayed.**
4. **Quantity selection: Select different quantities of an item (e.g., 1, 5, or 10) and verify that the correct quantity is reflected in the cart.**
5. **Product details: Verify that the added item in the cart displays the correct product information, such as name, price, and image.**
6. **Out of stock: Try adding an item that is out of stock and validate that an appropriate message is displayed, and the cart remains unchanged.**
7. **Invalid item: Attempt to add an item that does not exist in the product catalog and ensure that it is not added to the cart.**
8. **Cross-browser compatibility: Test the functionality of adding an item to the cart on different browsers (Chrome, Firefox, Safari, etc.) and verify consistent behavior.**

9. **Concurrent users: Simulate multiple users simultaneously adding items to the cart and confirm that each user's cart remains separate and unaffected by others.**
10. **Add-ons or options: Test adding an item with additional options or add-ons (e.g., size, color, customization) and verify that the selected options are correctly reflected in the cart.**

Q103: You are testing an e-commerce website that offers multiple payment options, including credit cards, PayPal, and bank transfers. During your testing, you encounter a scenario where a customer successfully places an order using a credit card, but the order status does not update in the system, and the payment is not recorded. How would you go about investigating this issue? Can you outline the steps you would follow to isolate the problem?

- **Reproduce the issue: Start by replicating the scenario where the problem occurred. Follow the same steps as the customer, including selecting the product, adding it to the cart, proceeding to the checkout process, and making the payment using a credit card. Note down any specific details or steps that could be relevant to the issue.**
- **Check for error messages: After making the payment, carefully review the website for any error messages or notifications related to the order status or payment process. Note down any error codes or messages displayed on the screen.**
- **Review system logs: Examine the system logs to gather information about the order and payment process. Look for any error logs, warnings, or exceptions related to the specific order and payment. Pay attention to timestamps, error codes, and any relevant log entries.**
- **Check payment gateway integration: Verify the integration with the credit card payment gateway. Ensure that the communication between the website and the payment gateway is functioning correctly. Review any documentation or specifications provided by the payment gateway service to understand the expected behavior and potential issues.**
- **Test with different credit cards: Attempt to reproduce the issue using different credit cards from various card issuers or networks. Determine if the problem is specific to a particular type of credit card or if it occurs consistently across different cards.**
- **Test other payment options: Verify if the issue is specific to credit card payments or if it affects other payment options as well, such as PayPal or bank transfers. Make test purchases using these alternate payment**

methods to check if the order status and payment recording work correctly.

- Contact payment gateway provider: Reach out to the payment gateway provider’s support or technical team. Describe the issue encountered and provide any relevant details, error messages, or log entries. Collaborate with them to investigate if there are any issues on their end or any specific configuration requirements.
- Check backend order processing: Inspect the backend order processing system to ensure that it is receiving and processing payment data correctly. Verify if there are any issues with data synchronization or communication between the website and the backend systems responsible for updating the order status and recording payments.
- Check database entries: Analyze the database entries related to the order and payment. Verify if the necessary data is being stored correctly, including payment details, timestamps, and order status. Compare the database entries with the expected results to identify any discrepancies.
- Collaborate with developers: Collaborate with the development team and share your findings, including error messages, log entries, and any specific details you have gathered. Work together to investigate the issue further and determine if it is a software bug, integration problem, or configuration issue.

By following these steps, you can systematically investigate the issue and gather relevant information to isolate the problem. Collaboration with different stakeholders, including support teams and developers, is crucial to ensure a comprehensive investigation and resolution of the issue.

Q104. When our release is in the next 2 hours & as a Tester, we found a blocker bug; then what will you do?

First thing, raise the bug; we need some ID to track for sure, then discuss with the Lead/Manager and call the development team to see if they can replicate and fix or have some workaround. If we have a workaround, it’s always suggested to go with it for now, as the fix may create new issues. If we don’t have then discuss it with the Product Owner and analyze the business impacts based on that either we need to fix it or notify the customer that it’s a known issue for this patch, and we fix it later and then deploy.

Q105. You have 30 test cases to execute and you have limited time to deliver, you cannot take another resource for help, so how you will execute these test cases to deliver on time? What will be your strategy?

In such a scenario, we can do:

- **Prioritize the test cases depending on risk, cost, time, and other factors**
- **Ask what would happen if you don't execute 50% of those 30 test cases**
- **Understand the 30 test cases and create a mental model to ascertain the best way to test rather than running through step by step for 30 test cases.**
- **Look for ways to automate at lower levels fully or partly for the 30 test cases.**
- **Ask yourself why we end up in this situation.**
- **Everybody has 24 hours, if it cannot be done in 24 hours, there is no way anybody can bring in the 25th hour, so be candid in your feedback.**
- **Skip 50% of executing 30 test cases, rest 50% monitor in production what those test cases would actually do and how it's behaving. Leverage unit test cases that can cover up for those 30 test cases**
- **Use feature toggle, where you release the product, test in production and release again with just a toggle once you are sure it works as expected.**
- **Use Blue–Green deployments.**
- **Highlight the risks to the stakeholders.**

Q106. You are testing a login page, and when you enter valid credentials, the page remains stuck with a loading spinner, and you cannot proceed further. What could be the possible reasons for this issue?

Possible reasons for the issue could be:

- **The server might be slow or unresponsive, causing the loading delay.**
- **There could be a problem with the backend authentication process.**
- **The JavaScript code responsible for handling the loading spinner might have errors.**
- **Network connectivity issues could be causing the delay in receiving the response.**

Q107. You are testing an e-commerce website, and after placing an order, the confirmation email is not sent to the user. How would you approach this problem?

To address this issue, I would follow these steps:

- **First, verify that the user's email address is correctly recorded during the order placement.**
- **Check the email server logs to see if any errors occurred during the email-sending process.**
- **Manually trigger the order confirmation email to see if there's a delay in the email delivery.**
- **Validate the SMTP settings and credentials to ensure the email server is configured correctly.**
- **Confirm if the email is not landing in the spam folder.**
- **Investigate if any recent code changes might have impacted the email functionality.**

Q108. You are testing a mobile app, and when you rotate the device from portrait to landscape mode, the app crashes. How do you troubleshoot this issue?

To troubleshoot the app crash on rotation issue, we would perform the following steps:

- **Reproduce the issue consistently by rotating the device multiple times.**
- **Check if the problem occurs with all devices or specific ones.**
- **Analyze the crash logs or error messages generated by the app.**
- **Examine the code that handles the orientation change to identify any bugs or unhandled exceptions.**
- **Verify if the app's layout and elements adjust correctly to the new orientation.**
- **Test the app on different versions of the operating system to see if the issue is OS-specific.**
- **Collaborate with developers to resolve the problem and retest the fix.**

Q109. You are testing a financial application, and when you enter negative values in a transaction, the application allows it, leading to incorrect calculations. How do you approach this issue?

To address the problem of allowing negative values in transactions, we would follow these steps:

- **Verify the application's business rules and requirements regarding transaction values.**

- **Test the application with valid positive values to ensure correct calculations.**
- **Attempt to enter negative values and verify if the application correctly restricts them.**
- **Inspect the server-side validation to see if it prevents negative values effectively.**
- **Check if any client-side scripts allow negative values to pass through.**
- **Communicate the issue to the development team with clear steps to reproduce and fix the problem.**

Q110. You are testing a web application, and when you navigate through different pages, the URL remains the same. How do you handle this situation?

In this case, we would apply the following actions:

- **Ensure that the expected functionality is not dependent solely on the URL changes, as modern web applications often use AJAX and single-page application (SPA) techniques.**
- **Confirm that essential data is not solely stored in the URL parameters.**
- **Use other elements like page content, breadcrumb navigation, or page titles to verify if the correct page is loaded.**
- **Check the usage of history. push State() or similar methods to modify the URL dynamically in SPA applications.**
- **If the issue persists and URL changes are indeed essential for the application, communicate the problem to the development team to investigate further.**

Q111. You re-testing a messaging app, and users report that sometimes messages are delivered to the wrong recipients. How do you approach this issue?

To tackle messages being delivered to the wrong recipients, we would undertake these steps:

- **Replicate the issue by sending messages to different users and verifying if they reach the intended recipients.**
- **Cross-check if the problem is device-specific or occurs on multiple devices and platforms.**
- **Analyze the server-side message delivery mechanism for any misrouting issues.**

- **Check the client-side code for any potential data handling or synchronization problems.**
- **Validate the user authentication and authorization processes to ensure data privacy and security.**
- **Collaborate with developers to identify and rectify the root cause.**

Q112. You are testing a video streaming service, and some users complain about buffering issues and playback interruptions. How do you troubleshoot this problem?

To troubleshoot the buffering and playback interruptions in the video streaming service, we would follow these steps:

- **Confirm if the issue is widespread or specific to certain videos or users.**
- **Check the internet connection speed and network stability during video playback.**
- **Evaluate the server's capacity and performance to handle concurrent streaming requests.**
- **Monitor the server logs for any error messages related to video delivery.**
- **Test the service on different devices and browsers to see if the problem is platform-specific.**
- **Check if the videos are appropriately optimized for streaming to reduce buffering.**
- **Collaborate with the development and infrastructure teams to optimize video delivery and address server-side bottlenecks.**

Q113. You are testing a healthcare app that stores sensitive patient data, and users report concerns about data privacy. How would you ensure the app complies with privacy regulations?

To ensure data privacy compliance in the healthcare app, we would follow these measures:

- **Verify that the app collects only the necessary patient data as per the defined requirements.**
- **Check if the app implements proper encryption techniques for storing and transmitting sensitive data.**
- **Test the app's authentication and authorization mechanisms to prevent unauthorized access to patient records.**
- **Validate if the app enforces proper user roles and access controls to limit data access to authorized personnel.**

- **Ensure that the app provides users with the option to consent to data collection and understand the privacy policy.**
- **Collaborate with the development team to address any identified security vulnerabilities promptly.**
- **Stay updated with relevant privacy regulations and guidelines to ensure continuous compliance.**

Q114. You are testing a gaming application, and users report that the game crashes randomly during gameplay. How do you troubleshoot this issue?

To troubleshoot the random game crashes, we would follow these steps:

- **Replicate the issue by playing the game under various conditions and scenarios.**
- **Check if the problem occurs on specific levels, actions, or devices.**
- **Monitor the system resources (CPU, memory, GPU) during gameplay to identify resource-intensive areas.**
- **Analyze the game logs or error reports generated upon each crash.**
- **Validate if the crashes are consistent with certain actions or patterns.**
- **Collaborate with the game developers to identify any bugs or memory leaks that could be causing the crashes.**
- **Test the game on different devices and operating systems to verify if the issue is platform-specific.**

Q115. You are testing a social media app, and users report that they sometimes receive notifications for activities they did not perform. How do you handle this situation?

To address the issue of incorrect notifications, we would take the following steps:

- **Verify if the notifications received by users match their actual activities on the app.**
- **Check if the issue is device-specific or occurs on multiple devices and platforms.**
- **Analyze the server-side notification generation and delivery mechanism for any misrouting issues.**
- **Validate the user authentication and authorization processes to ensure only relevant notifications are sent.**
- **Confirm if there are any server-side caching issues causing delayed or wrong notifications.**