BCA SOFTWARE TESTING

QUESTION BANK

PART A :

1. **What is software testing?**
   - **Explanation:** Software testing is the process of evaluating a software application to identify any discrepancies between expected and actual results. The goal is to ensure that the software meets the specified requirements and works correctly.

2. **What is the difference between verification and validation?**
   - **Explanation:** Verification focuses on checking if a product conforms to specified requirements, while validation involves evaluating the system during or at the end of the development process to ensure it satisfies the intended use.

3. **What is the purpose of test cases?**
   - **Explanation:** Test cases are designed to verify the functionality, performance, and other aspects of a software application. They serve as a set of instructions to guide testers in executing tests to uncover defects or errors.

4. **What is regression testing?**
   - **Explanation:** Regression testing ensures that changes to the codebase (new features, bug fixes) do not adversely affect existing functionalities. It involves re-executing previously executed test cases on the modified code.

5. **Explain the difference between white-box testing and black-box testing.**
   - **Explanation:** White-box testing involves testing internal structures or workings of a software application, while black-box testing focuses on testing the functionality without considering the internal code.

6. **What is the significance of the boundary value analysis technique?**
   - **Explanation:** Boundary value analysis tests input values at the boundaries of allowed ranges. It helps identify potential errors at the edges of input domains, where errors are more likely to occur.

7. **Define test plan and test strategy.**
   - **Explanation:** A test plan outlines the scope, approach, resources, and schedule of testing activities. A test strategy defines the overall testing approach, including the testing types, levels, and entry/exit criteria.

8. **What is the purpose of the Defect Life Cycle?**
   - **Explanation:** The Defect Life Cycle describes the various stages a defect goes through, from discovery to closure. It includes stages such as identification, logging, fixing, retesting, and closure.

9. **Explain the concept of equivalence partitioning.**

- **Explanation:** Equivalence partitioning involves dividing input data into classes or partitions and selecting representative values from each partition for testing. This technique reduces the number of test cases while ensuring comprehensive coverage.

10. **What is the difference between static testing and dynamic testing?**
    - **Explanation:** Static testing involves reviewing and analyzing the code without executing it, while dynamic testing involves running the code to identify errors or defects during runtime.

11. **Define the term "Test Harness."**
    - **Explanation:** A test harness is a set of tools, libraries, and software components designed to facilitate the execution of tests. It provides an environment for running test cases and collecting results.

12. **What is usability testing?**
    - **Explanation:** Usability testing evaluates a software product's user interface and overall user experience to ensure it is user-friendly. It assesses factors such as ease of use, efficiency, and user satisfaction.

13. **Explain the purpose of a traceability matrix.**
    - **Explanation:** A traceability matrix links requirements to test cases, ensuring that each requirement is tested. It helps in tracking the completeness of testing and ensures that all specified requirements are covered.

14. **What is the importance of performance testing?**
    - **Explanation:** Performance testing evaluates a system's responsiveness, scalability, and stability under various load conditions. It helps identify and address performance-related issues, ensuring the software performs well in real-world scenarios.

15. **What is the role of a test environment in software testing?**
    - **Explanation:** A test environment is a setup that mimics the production environment where testing is carried out. It includes hardware, software, network configurations, and data. A stable and representative test environment is crucial for accurate testing results.

16. **What is the purpose of smoke testing?**
    - **Explanation:** Smoke testing, also known as build verification testing, is conducted to check whether the essential functionalities of a software application are working correctly after a new build or release. It helps determine if further testing can proceed.

17. **Explain the concept of risk-based testing.**
    - **Explanation:** Risk-based testing involves prioritizing testing efforts based on the areas of the software that pose the highest risk. This approach ensures that testing focuses on critical functionalities and potential points of failure.

18. **What is the difference between validation testing and system testing?**
    - **Explanation:** Validation testing checks if the software meets the customer's requirements, while system testing verifies the entire system's behavior, ensuring all components work together as intended.

19. **Define the term "Monkey Testing."**
    - **Explanation:** Monkey testing involves random and exploratory testing without specific test cases. Testers interact with the system in an unplanned manner to discover unforeseen issues.

20. **What is the purpose of the bug life cycle?**
    - **Explanation:** The bug life cycle outlines the stages a defect goes through, including identification, assignment, fixing, retesting, and closure. It helps manage the process of resolving and verifying defects.

21. **Explain the concept of positive testing.**
    - **Explanation:** Positive testing involves validating that a system behaves as expected with valid inputs. It checks if the application functions correctly under normal or expected conditions.

22. **What is the significance of code coverage in testing?**
    - **Explanation:** Code coverage measures the percentage of code executed during testing. It helps assess the thoroughness of testing and identifies areas of code that remain untested.

23. **Define the term "Test Oracle."**
    - **Explanation:** A test oracle is a mechanism or principle used to determine whether the output of a test is correct. It may involve using specifications, previous versions, or expert judgment to validate results.

24. **What is the purpose of the Traceability Matrix in test execution?**
    - **Explanation:** During test execution, the Traceability Matrix helps track the progress of testing by linking test cases to requirements. It ensures that all specified requirements are covered and tested.

25. **Explain the concept of ad-hoc testing.**
    - **Explanation:** Ad-hoc testing is informal and unplanned testing without predefined test cases. Testers explore the application freely, trying to identify defects without following a structured approach.

26. **What is the role of a test script in automated testing?**
    - **Explanation:** A test script in automated testing is a set of instructions written in a programming language to perform specific test actions. It enables the automation tool to execute test cases automatically.

27. **Define the term "Sanity Testing."**
    - **Explanation:** Sanity testing, also known as build verification testing, is a quick, focused testing effort to ensure that specific functionalities work as intended after changes or fixes.

28. **What is the importance of security testing in software development?**
    - **Explanation:** Security testing assesses a system's resistance to unauthorized access, vulnerabilities, and potential threats. It helps identify and address security issues, protecting the software from malicious activities.

29. **Explain the concept of non-functional testing.**
    - **Explanation:** Non-functional testing evaluates aspects such as performance, usability, reliability, and security of a software application. Unlike functional testing, it focuses on characteristics that do not involve specific functionalities.

30 **What is the purpose of a test log in software testing?**
    - **Explanation:** A test log records the details of test execution, including test case results, issues encountered, and any deviations from the expected behavior. It serves as a valuable reference for test reporting and analysis.

31. **Explain the V-Model in software testing. How does it differ from the waterfall model, and what advantages does it offer in terms of testing?**
    - **Explanation:** The V-Model is a software development and testing process that emphasizes the relationship between each development phase and its corresponding testing phase. It is an extension of the waterfall model, with testing activities integrated at each stage. The V-Model offers advantages such as early detection of defects, better traceability, and a systematic approach to testing.

32. **Discuss the challenges and benefits of implementing test automation in a software development project. Provide examples of scenarios where automation is suitable and where manual testing might be preferred.**
    - **Explanation:** Test automation can improve efficiency, repeatability, and coverage in testing. However, challenges include initial setup costs, maintenance efforts, and the need for skilled automation engineers. Scenarios suitable for automation include regression testing, performance testing, and large-scale projects. Manual testing may be preferred in exploratory testing or for scenarios requiring human intuition.

33. **Elaborate on the principles of risk-based testing. How can a project team effectively identify, assess, and prioritize risks in the context of testing?**
    - **Explanation:** Risk-based testing involves identifying, assessing, and managing risks to prioritize testing efforts. The principles include focusing on high-risk areas, allocating more testing resources to critical functionalities, and adapting the test strategy based on project changes. Techniques like risk analysis, brainstorming, and historical data analysis can help identify and assess risks.

34. **Compare and contrast the concepts of verification and validation in the context of software testing. Provide examples to illustrate the application of these concepts in a real-world scenario.**
    - **Explanation:** Verification ensures that the software is built correctly according to specifications, while validation ensures that the software meets the customer's requirements. An example of verification is code reviews, ensuring adherence to coding standards. Validation involves activities like user acceptance testing, ensuring the software meets user expectations.

35. **Examine the importance of test planning in the software testing life cycle. Provide a step-by-step guide on how to create a comprehensive test plan for a software project.**
    - **Explanation:** Test planning is crucial for defining the scope, objectives, resources, and schedule of testing activities. A comprehensive test plan includes sections on test strategy, test scope, resource requirements, schedule, test deliverables, and exit criteria. It also outlines the test approach, test design, and test execution processes.

36. **Discuss the concept of usability testing. How can usability testing contribute to the overall quality of a software product, and what methodologies can be employed in conducting usability tests?**
    - **Explanation:** Usability testing evaluates the user-friendliness of a software product. It assesses factors such as ease of use, efficiency, and user satisfaction. Usability testing contributes to overall quality by ensuring a positive user experience. Methodologies include task-based testing, heuristic evaluation, and user surveys.

37. **Explain the principles of static testing. Provide examples of static testing techniques and discuss how they can be applied to improve the quality of software artifacts.**
    - **Explanation:** Static testing involves reviewing and analyzing software artifacts without executing them. Techniques include code reviews, inspections, and walkthroughs. These techniques help identify defects early in the development process, reducing the cost of fixing issues later.

38. **Define and discuss the concept of traceability in software testing. How does traceability contribute to the effectiveness of the testing process, and what challenges might be encountered in maintaining traceability?**
    - **Explanation:** Traceability involves linking requirements to test cases, ensuring comprehensive coverage. It contributes to the effectiveness of testing by providing a clear understanding of which requirements have been tested. Challenges in maintaining traceability include changes in requirements, evolving project scope, and managing the traceability matrix.

39. **Elaborate on the key components and objectives of performance testing. Discuss various types of performance testing and how they address different aspects of a software application's performance.**
    - **Explanation:** Performance testing evaluates a system's responsiveness, scalability, and stability. The key components include load testing, stress

testing, and scalability testing. Load testing assesses performance under expected loads, stress testing checks system behavior under extreme conditions, and scalability testing evaluates the system's ability to handle growing user loads.

40. **Examine the role of a test environment in the software testing process. What considerations should be taken into account when setting up and maintaining a test environment for a project?**
    - **Explanation:** A test environment provides the infrastructure for executing tests. Considerations include hardware, software, configurations, and data. The environment should mirror the production environment to ensure accurate testing results. Challenges may include maintaining consistency between environments and managing dependencies.

41. **Discuss the principles and challenges of security testing in software development. Provide examples of common security testing techniques and explain how they contribute to identifying and mitigating security vulnerabilities.**
    - **Explanation:** Security testing assesses a system's resistance to unauthorized access and vulnerabilities. Techniques include penetration testing, vulnerability scanning, and security code reviews. Challenges include the evolving nature of security threats and the need for continuous monitoring.

42. **Explain the concept of model-based testing. How can models be used to design and execute test cases, and what advantages does model-based testing offer over traditional test design methods?**
    - **Explanation:** Model-based testing uses models to design and generate test cases. Models represent the system's behavior, inputs, and outputs. Model-based testing offers advantages in terms of efficiency, coverage, and maintaining synchronization between specifications and test cases.

43. **Evaluate the impact of defects on the software development life cycle. How can defect prevention strategies be integrated into the development process, and what role does early defect detection play in reducing development costs?**
    - **Explanation:** Defects can lead to increased costs, delays, and compromised product quality. Defect prevention strategies include code reviews, static analysis, and training. Early defect detection reduces rework costs and prevents defects from propagating to later stages of development.

44. **Discuss the challenges and benefits of adopting agile methodologies in software testing. How does agile testing differ from traditional testing approaches, and what practices can enhance the effectiveness of testing in agile projects?**
    - **Explanation:** Agile methodologies emphasize iterative development and collaboration. Challenges include adapting to changing requirements and timelines. Benefits include faster time-to-market and improved customer satisfaction. Agile testing involves continuous testing, test-driven development, and close collaboration between development and testing teams.

45. **Examine the concept of dynamic testing in software testing. Provide examples of dynamic testing techniques and explain how they are applied to ensure the correctness and reliability of a software application.**

- **Explanation:** Dynamic testing involves executing software to uncover defects. Techniques include functional testing, integration testing, and system testing. Functional testing ensures that each function of the software operates as expected, while integration testing verifies the interactions between different components. System testing evaluates the entire system's behavior against specified requirements.

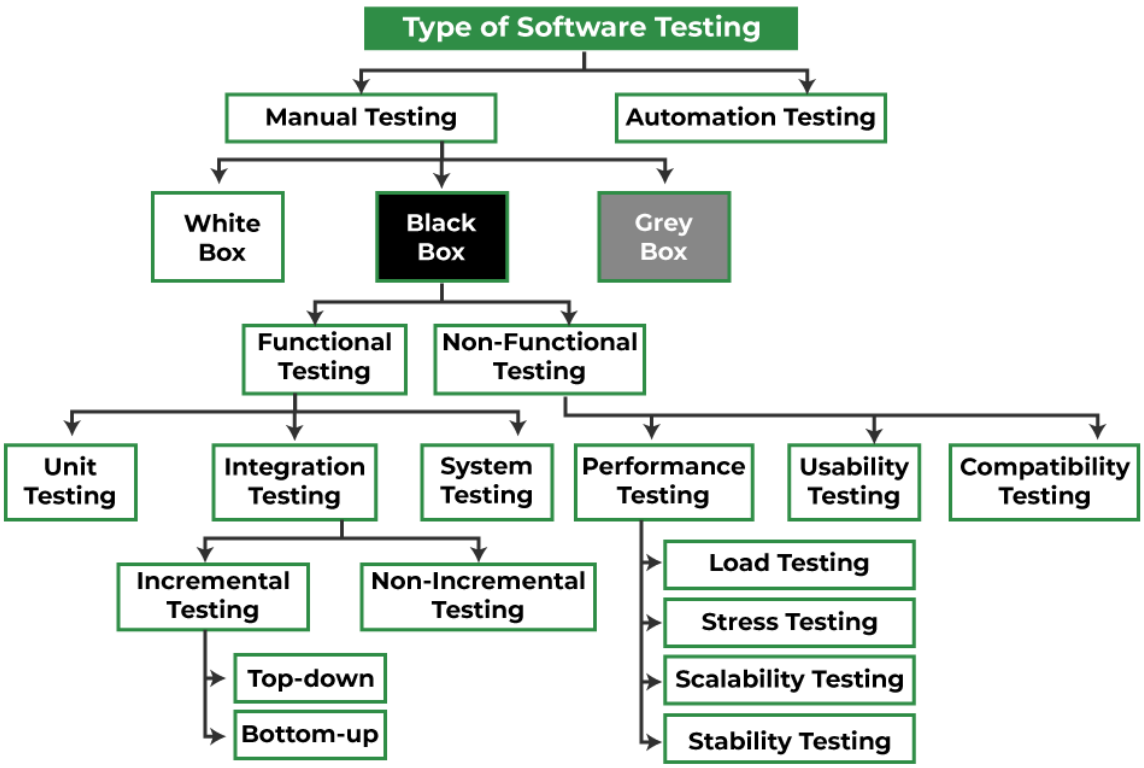
PART B :

1. What is Software Testing?

Software testing is the process of evaluating and verifying if a software product does what it is supposed to do. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

2. What are the different types of software testing?

There are many ways to test software. Some types of testing are conducted by software developers while some are conducted by specialized quality assurance staff. Here are the different types of software testing with brief descriptions for each.

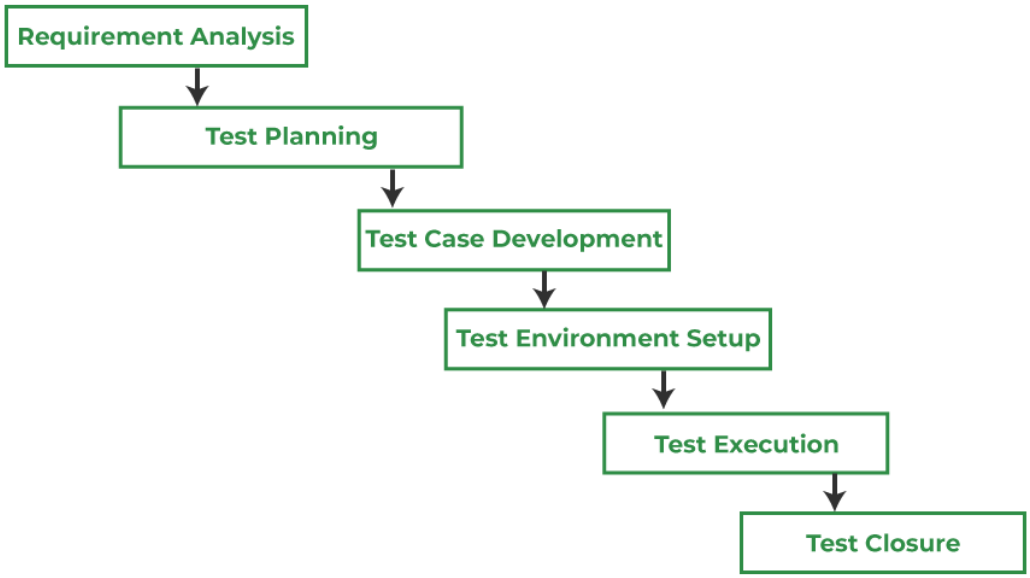


Testing Type	Description
Manual Testing	**Testing any software according to the client's needs without using any automation tool is known as Manual Testing.**
Automation Testing	**Testing any software according to the client's needs using an automation tool is called Automation Testing.**
White Box Testing	**In white-box testing, internal structure, internal design, the data structure used, the code structure of the software, and the working of the software are analyzed.**
Black Box Testing	**In Black box testing, the functionalities of the software are tested without having knowledge of the internal code structure, internal paths, and implementation details of the software.**
Grey Box Testing	**In grey box testing, there is partial knowledge of the internal structure of the application. The purpose is to search and identify the defects due to improper code structure or improper use of applications.**
Functional Testing	**In functional testing, the software system is validated against the functional requirements.**
Non-Functional Testing	**In non-functional testing, the non-functional parameters like reliability, load test, performance, and accountability of the software are tested.**
Unit Testing	**Unit testing is the type of software testing where the individual units or modules of the system are tested.**
Integration Testing	**Integration testing is the type of testing where software modules are integrated logically and tested as a group.**
System Testing	**This type of testing validates the sully integrated software product.**
Performance Testing	**This testing is used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload.**
Usability Testing	**It is also known as User Experience Testing. It is a testing type to check how easy and user-friendly a software application is.**

Compatibility Testing	**Compatibility testing is a type of non-functional testing that is done to check whether the software is capable of running on different hardware, operating systems, applications, network environments, or mobile devices.**
Incremental Testing	**Incremental testing involves integrating modules one by one to uncover defects by developers.**
Non-Incremental Testing	**In non-incremental testing, the data is created in one module and is combined with all the other modules to check and test the flow of data between them.**
Top-Down Testing	**In top-down testing, the higher-level modules are tested first then the lower-level modules are tested. In this approach, the subs are used as a replacement for the submodules, if the invoked submodule is not developed.**
Bottom-Up Testing	**In bottom-up testing, the lower-level modules are tested first, then the higher-level modules are tested. This approach uses test drivers which are mainly used to pass the required data to the sub-modules means from the higher level to the lower level module.**
Load Testing	**In this type of testing, the performance of the software application is tested under a specific expected load.**
Stress Testing	**It is a type of testing that verifies the stability and reliability of the software.**
Scalability Testing	**It is a type of testing method that measures the performance of the system or network when the number of user requests is scaled up or down.**
Stability Testing	**This type of testing is done to verify whether the application can continuously perform well or just above the acceptable period.**

3. What are the different phases of the software testing life cycle?

There are six phases in the software testing life cycle:



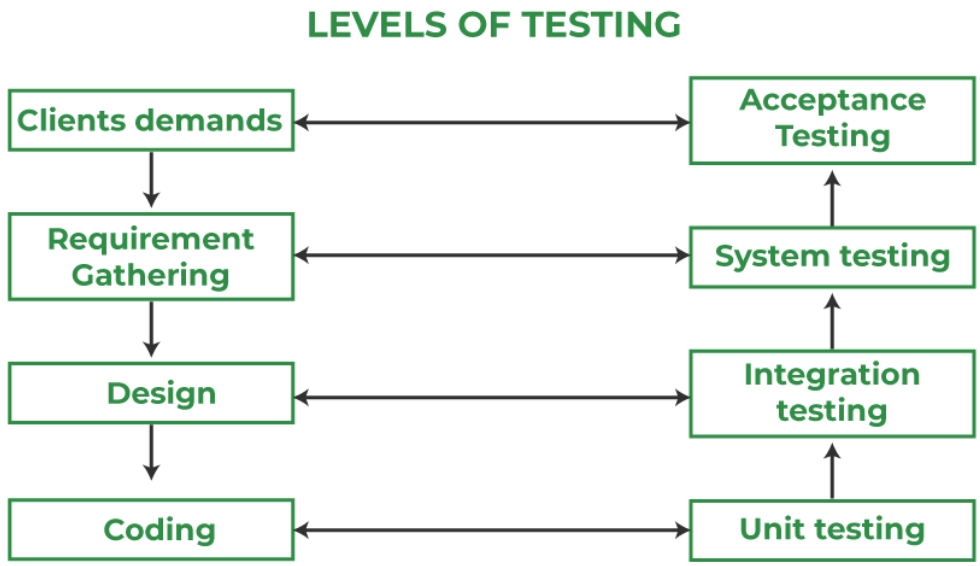
1. **Requirements Analysis-** In this phase, the testing team studies the requirements to identify the testable requirements. The requirements can be functional or non-functional. The various activities in this phase are-
    - Identify the type of test to be conducted.
    - Prepare Requirement Traceability Matrix.
    - Identify test environment details.
    - Gather information about testing priorities.
2. **Test Planning-** Senior QA manager decides test plan strategy along with cost estimates and efforts for the project. In this phase, the test environment, test schedule, and test limitations are also determined. This phase includes the following activities-
    - Preparation of the test strategy document.
    - Test Effort and cost estimation.
    - Test limitation, test schedule determination.
    - Resource planning.
    - Determining roles and responsibilities.
3. **Test Case Development-** This phase involves the creation, verification, and rework of test cases after the test plan is ready. The following activities are included in this phase-
    - Test case creation.
    - Review test cases.
    - Create test data.
4. **Test Environment Setup-** In this phase, the hardware and the software conditions under which a work product is tested are determined. This phase includes the following activities-
    - Understand the environment setup and determine the hardware and software requirements for the test environment.
    - Set up the test environment.
    - Setup test data.
5. **Test Execution-** In this phase, the testers carry out the testing of the software build based on the prepared test plans and test cases. The bugs reported here are reverted back to the development team for correction and retesting will be carried out. The activities involved in this phase are-

- Test script execution.
- Test script maintenance.
- Bug reporting.
- Map defects to test cases in RTM.
- Track defects to closure.

6. **Test Closure-** This is the last phase of test execution which involves several activities like test completion reporting, and collection of test results. This phase involves the following activities-
    - Analyze testing artifacts to identify strategies.
    - Prepare test metrics.
    - Document learning of the project.
    - Prepare test closure project.

4. What are the different levels of testing?

There are mainly four levels of software testing-



1. **Unit Testing-** The aim of this level of testing is to test each individual unit or module of the software by separating it. In this type of testing, it is checked whether the components are fulfilling the functionalities or not.
2. **Integration Testing-** In this type of testing, the different modules are combined and tested as a group to make sure that the integrated system is ready for system testing.
3. **System Testing-** In this type of testing, the complete, integrated system is tested. The system's compliance is checked as per the requirements. The overall interaction of the components is tested.
4. **Acceptance Testing-** It is usually done by the user or customer. However, other stockholders also are involved in this process. It is used to evaluate whether the requirements are met as per its delivery. This is also known as User Acceptance Testing(UAT).

5. What is use case testing?

It is a testing technique that helps to identify the test cases to cover the entire system from start to end. Test cases are the transactions that are performed between the user and the system. The use case testing helps in the following ways-

- Identify the gaps in the software application.
- It helps to capture the functional requirements of the system.
- It helps to manage the complexity since it focuses on one specific usage aspect at a time.

6. What is a traceability matrix?

A traceability matrix also known as Requirement Traceability Matrix(RTM) is a document that is used in the development of the software application to trace the requirements proposed by the client to the system being built. Some of the features of RTM are-

- It is prepared before test execution to make sure that all the requirements are covered in the test cases.
- The test engineers will prepare RTM for the respectively assigned modules.
- These will be submitted to the Test Lead.
- Test Lead will verify the specific module RTM to see if all the requirements of that module are mapped to corresponding test cases and in the end, will consolidate all the RTMs and prepare one necessary RTM document.
- The main purpose of RTM is to check that all requirements are checked via test cases.

7. Why is software testing important?

Some of the reasons why software testing is important are-

- **Increase the quality-** The quality of the software can be determined by the number of defects identified during the testing process. Those defects can be fixed with the software development life cycle. Testing ensures a quality product is delivered to the customer.
- **Reduce risks-** Defects discovered during the testing process should either be fixed or removed from the final product to ensure the seamless execution of the software during the live operation. Continuous testing is important to mitigate the risks.
- **Security-** If a specific product has undergone thorough and continuous testing then the user can be ensured that a reliable product will be delivered to them. The personal credentials of the user can be considered to be safe.
- **Satisfaction of the customer-** System stability is what a customer really wants. Continuous testing throughout the software development process ensures that stable software has been developed, therefore increasing the confidence in the customer once the software will be released into the live environment.
- **Cost-Effective-** The cost-effectiveness of the software is one of the top reasons why software testing is such an important activity in the software development process. Testing early helps the project managers to have better control of the budget of software. Discovering defects and fixing them in the earlier stages not only enables the system to be better but also reduces unexpected costs.
- **Enhancing the development process-** The testing team should work in parallel with the development team, which is useful for the acceleration of the development process.
- **Determine the performance of the software-** Software testing is the easiest way to determine the performance of the software. If the software has low performance, then it will bring down the reputation of the company. Thus, conducting continuous testing will help to uncover errors in the software and thus increase the performance of the software.

8. What is a test case?

A test case is a document specifying the test data, precondition, expected results, and post-conditions that are developed for a particular test scenario to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine whether the different features in the system are performing as expected. Writing test cases can help to reveal errors or bugs in the system. The benefits of an effective test case include the following-

- Good test coverage.
- Reusable test cases.
- Helps to avoid training for every new test engineer.
- Improved quality of the software.
- More satisfied customers.

The typical test case parameters are-

- Test Case Name/ ID
- Test Case Type
- Requirement Number
- Module
- Severity
- Status
- Release
- Version
- Pre-condition
- Test Data
- Summary

9. What does a test plan consist of?

According to IEEE 829 test plan standard, the test plan consists of the following headings-

- **Test plan identifier-** It identifies the project and includes information about the version and test plan type.
- **Introduction-** It contains the summary of the test plan. It sets objectives, scope, goals, resources, and budget constraints.
- **Test items-** It lists the artifacts to be tested.
- **Features to be tested-** This section lists all the features and the functionality to be tested. It also contains a reference to the requirements specification document.
- **Features not to be tested-** This section lists the features and functionalities that are out of the scope of testing.
- **Approach-** The approach for testing contains details of how testing will be done, sources of test data, inputs, and outputs, testing techniques, and priorities.
- **Item pass/ fail criteria-** In this section, the success criteria for evaluation of the test results of each functionality in detail will be listed.
- **Suspension criteria and resumption requirements-** This section lists the criteria that may result in suspending the testing activities and subsequently also lists the requirements to resume the testing process.
- **Test deliverables-** These are the documents that will be delivered by the testing team at the end of the testing process. It may include test cases, sample data, test reports, and issue logs.
- **Testing tasks-** This section lists the testing tasks that may include describing dependencies between any tasks, resources required, and completion time for tasks.

- **Environmental needs-** This section lists the requirements for the test environment. It includes hardware, software, or any other environmental requirement for testing.
- **Responsibilities-** In this section, the roles and responsibilities that are assigned to the testing team are listed.
- **Staffing and training needs-** This section describes the training needs of the staff for carrying out the planned testing activities.
- **Schedule-** The schedule is created by assigning days and dates to the testing activities.
- **Risks and contingencies-** The test plan should contain the mitigation techniques for the identified risks, and should also include the contingencies.
- **Approvals-** This section contains the signatures of approvals from all the stakeholders.

10. What do you mean by Quality Assurance(QA)?

Quality assurance is defined as a procedure to ensure whether a software product meets specified requirements or not. It focuses on improving the quality of the software development process and making it efficient and effective as per the quality standards. The five major quality assurance functions are-

- **Technology transfer-** This involves getting the product design document as well as trial and error data and its evaluation.
- **Documentation-** This function controls the distribution and archiving of the documents. For incorporating any change in the document, a proper change control procedure needs to be adopted.
- **Validation-** In this function, the validation master plan for the system, and approval of test criteria for validating the product and process are also set.
- **Assuring quality products-** This function aims at checking and assuring the quality of the software product.
- **Quality improvement plans-** In this function, quality improvement plans are generated to check whether the software product meets the requirements.

11. Explain the Waterfall and Agile Models.

Software testing comes into the picture at different times in different software development methodologies. The two main software development methodologies are the Waterfall Model and Agile Model.

- In Waterfall Model, the requirements are gathered first. Then the next step is to create a specification document that drives the design and development phases. Finally, testing is conducted at the end of the software development life cycle once the complete software system is built.
- In Agile Model, testing is conducted in parallel as the software is getting built. The developers build a small functionality according to the requirements. The testers test this and acquire customer feedback which drives the further development of the software.

12. What is a user story?

In software development, the user story is an informal natural language general explanation of a software feature written from the perspective of the end-user. It is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. Some of the features of the user story are-

- It describes the type of user, what the user wants, and why.

- It helps to create a simplified description of a requirement.
- They are often recorded on index cards, on Post-it notes, or in project management software.
- User stories may be written by various stakeholders such as clients, users, managers, or development team members.
- They provide a user-focused framework for daily work.

13. What is a test environment?

The test environment comprises hardware and software configuration that allows testing teams to run test cases. It is set up according to the Application Under Test. The test bed may be a mix of the test environment and the test data with which it interacts. It involves setting up various distinct areas like-

- Setup of Test Server.
- Network.
- Test PC Setup.
- Bug Reporting.

14. What is Cookie Testing?

Cookie testing is a form of software testing where cookies produced in the web browser are examined. A cookie stores user information that can be used to track users' website navigation and can be used to communicate between different web pages. It is very important to keep a check on the cookies, and how they are written and saved in the system in order to avoid any security threats.

15. What is Bottom-up Testing?

It is a type of integration testing that tests the lowest components of a code base first. In these, low-level modules are tested first, and then the high-level modules are tested. The driver is a temporary module that is used to simulate the caller module for a module to be tested, to call the interface procedures of the module to be tested, and to report the results.

16. What is Dynamic Testing?

Dynamic testing is a software testing technique where the dynamic behavior of the code is checked. The purpose of this type of testing is to check and analyze the software behavior with dynamic variables and find the weak areas in the software runtime environment.

- In this type of testing, the software must be actually compiled and run.
- Giving the input values and checking if the output is as expected by executing specific test cases.

17. What is Risk-Based Testing?

Risk-Based Testing is a software testing technique that is based on the probability of the risks and involves assessing the risk based on the factors like:

- Software complexity.
- Criticality of the businesses.
- Frequency of use.
- Possible areas of defects.

It uses risks to prioritize the appropriate tests during test execution. It starts early in the software project, identifying the risks to system quality and using the knowledge of risks identified to guide testing planning, specification, preparation, and execution.

18. What is Fuzz Testing?

Fuzz testing or fuzzing is a software testing approach that injects invalid or malformed inputs into the system to reveal software vulnerabilities in an attempt to make it crash.

- It is carried out using automated software that generates new test cases and feeds the program inputs.
- It is used to detect if the program crashes or reveals other vulnerabilities.

19. What is Test Harness?

Test Harness is a collection of stubs, drivers, and other supporting tools that are required to automate test execution. It allows for the automation of tests.

- It contains all the information that is required to compile and run a test. For example, test cases, stubs, drivers, Source files under tests, etc.
- It helps to enhance the quality of the software components and applications.
- It helps developers to measure code coverage at a code level.

20. What is Concurrency Testing?

Concurrency Testing also known as Multi-user Testing checks the software performance when multiple users are logged into the system and perform actions simultaneously.

- It helps to monitor the effect on the system when multiple users are performing the same action at the same time.
- It helps to monitor the system for deadlocking, locking, and single-threaded code.

21. What is Defect Age?

Defect Age is defined as the time difference between the defect detected date and the current date provided the defect is still in the open state. It is divided into two parameters:

- **Defect Phase:** The defect phase is the numerical value that is assigned to a defect occurring at any phase, depending upon the degree of risk involved in the defect.
- **Defect Age Time:** Defect age time is a way to determine the difference between the date of defect detection and the time till when the defect is open or has been resolved.

22. What is Un-Installation Testing?

Uninstallation testing is a type of software testing that is performed to make sure that all the components of the application are removed during the process or not.

23. What is Equivalence Class Partition (ECP)?

Equivalence Class Partitioning is a black-box software testing technique that divides the input data into partitions of equivalent data from which test cases can be derived.

- In this approach, test cases are designed to cover each partition at least once.
- It divides the input data of software into different equivalence data classes.

24. What is Software Configuration Management?

Software Configuration Management (SCM) is a process to manage, organize, and control the changes in the code, document, and other entities during Software Development Life Cycle (SDLC).

- It is commonly used in software development groups in which several developers are concurrently working on a common set of files.

- SCM is designed to avoid the problem of sharing files in a multiuser environment.

25. What is a Test Script?

Test scripts are step-by-step instructions containing information about the system transactions that should be performed to validate the application or system under test.

- These are the programs that run tests on the software product/ application.
- The tester has to write and run test scripts to validate if the application's outcome meets the business requirements.

26. What is Test Bed?

Test Bed is a platform for conducting rigorous, transparent, and replicable testing that consists of specific hardware, software, operating system, network configuration, software configuration, etc.

27. What is Sanity Testing?

Sanity Test also known as Surface Testing is a type of software testing that is performed to make sure that the code changes made are working properly without any bugs.

- This type of testing is done on the stable build of the software.
- It is a subset of Regression testing.
- Sanity testing is usually done after the software product has passed the Smoke test.
- The focus of this type of testing is to validate the functionality of the application and not detailed testing.

28. What is Test Closure?

Test Closure is a document that provides a summary of all the tests that are covered in the software development lifecycle.

- It includes various activities like test completion reports, test completion matrix, a summary of test results, etc.
- The test closure process helps the other team members to get to know about the end of the testing process.
- It provides a complete report of all the errors discovered and resolved so that the source of the error can be located and resolution can be provided.

29. What is a Stub?

A stub is a small piece of code that is used during Top-down Integration Testing that takes the place of another component during testing. These act as a temporary replacement for the called module and give the same output as that of the actual product.

30. What is a Driver?

Test Driver is a small piece of code that is used during Bottom-up Integration Testing that simulates the behavior of the upper-level modules that are not yet integrated. These act as a temporary replacement for the calling module and give the same output as that of the actual product.

31. What is Cause-effect Graph?

Cause-effect Graph is a black box testing technique that is used to represent the relationship between a given outcome and all the factors that influence the outcome. It is based on the collection of requirements and used to determine the minimum possible test cases which can

cover a minimum test area of the software. The main advantage of this testing technique is that it reduces the test execution time and cost.

32. What is Test Strategy?

Test Strategy is a document that outlines the testing technique that is used in the Software Development Life Cycle and guides QA teams to define Test Coverage and Testing scope

33. What is Test Scenario?

A test Scenario is a detailed document that covers end to end functionality of a software application that can be tested. It is also known as Test Possibility or Test Condition. In this, the testers need to put themselves in the place of the user as they test the software application from the user's point of view.

34. What is Code Coverage?

Code Coverage is a software testing metric that is used to determine how much of the code is tested. This helps in assessing the quality of the test suite and analyzing how comprehensively a software is verified. It is one form of white box testing which finds the areas of the program not exercised by a set of test cases.

35. Explain the role of testing in software development.

Testing plays a vital role in software development. The techniques used for software testing differ from one company to another. Below are some of the important roles played by testing in software development-

- Examine the code for discovering problems and errors early in the system.
- Testing evaluates the capabilities of the program over the entire product.
- Testing helps in reviewing requirements and design as well as executing the code.
- Testing is important in measuring the system functionality and quality of the product.
- Testing plays a vital role in lowering the maintenance cost of the software product.
- It helps in providing interaction between developers and users.

36. What is a bug report?

During testing, the tester records all the observations, and other information useful for the developers or the management. This test record is known as a bug report. The bug report helps the team member in the following aspects-

- Understand the problem.
- Steps to reproduce the problem.
- The environment under which the defect or bug happens.
- The resolution is when the developer fixes the bug.

The few bits of information the bug report should contain are-

1. **Defect/ Bug Name-** A short headline describing the defect. It should be specific and accurate.
2. **Defect/ Bug ID-** Unique identification number for the defect.
3. **Defect Description-** Detailed description of the bug including the information of the module in which it was detected. It contains a detailed summary including the severity, priority, expected results vs actual output, etc.
4. **Severity-** This describes the impact of the defect on the application under test.

5. **Priority-** This is related to how urgent it is to fix the defect. Priority can be High/ Medium/ Low based on the impact urgency at which the defect should be fixed.
6. **Reported By-** Name/ ID of the tester who reported the bug.
7. **Reported On-** Date when the defect is raised.
8. **Steps-** These include detailed steps along with screenshots with which the developer can reproduce the same defect.
9. **Status-** New/ Open/ Active
10. **Fixed By-** Name/ ID of the developer who fixed the defect.
11. **Data Closed-** Date when the defect is closed.

37. What is the purpose of risk-based testing?

Risk-based testing involves accessing the risk based on the software complexity, frequency of use, and many other factors. It prioritizes testing of the functionality and features of the system which are more impactful and are likely to have defects. The purpose of risk-based testing is to-

- Identify risks to system quality.
- Use the knowledge of risk to guide testing planning, specification, preparation, and execution.
- It involves both mitigation and contingency. Mitigation here means to test the reduce the

38. What are the different types of testing metrics?

The different types of testing metrics are-

- **Process Metrics-** These metrics are essential for the improvement and maintenance of the process in SDLC. These are used to improve the process efficiency of the SDLC.
- **Product Metrics-** These define the size, performance, quality, and complexity of the product. It deals with the quality of the software product and thus can help developers to enhance the software product quality.
- **Project Metrics-** These define the overall quality of the project. These can be used to measure the efficiency of the project team or any testing tools being used by the team members.

39. What do you mean by Defect Cascading?

Defect Cascading is when one defect leads to the discovery of another defect by software testers. There are several reasons behind defect cascading but one of the main reasons is it occurs because the original defect was not fixed properly.

40. What is Test-Driven Development?

Test-Driven Development is the software development approach in which test cases are created for each functionality and tested first and if the test fails then a new code is written to pass the test, thus making the code bug-free. It is an iterative approach that involves combining the two processes, the creation of test cases, and refactoring. Some of the benefits of test-driven development are-

- **Cleaner and better design code-** TDD helps in better design decisions as it helps to understand how the code will work and how it will interact with other modules. It allows the writing of more maintainable and smaller codes.

- **Early bug detection-** Through test cases, developers can test the functionality and if the test fails then a new code can be written. Thus, bug-free code can be achieved since bugs are detected early.
- **Good for developers-** Although developers have to spend extra time in writing the test cases eventually it will take a lot less time for debugging and developing the new features.
- **Confidence to Refractor-** In the case of code refractors, there are chances of breaks in the code. With the help of a set of automated test cases, a proper warning can be given and breaks can be fixed before release.
- **Extensible code-** TDD can result in an extensible code with fewer bugs that can be updated with minimal risks.

41. What is the difference between verification and validation in Testing?

S No.	Verification	Validation
1	Verification is the process of checking whether the software achieves its goals without any bugs.	Validation is the process of checking whether the software product has high-level requirements.
2	Verification is static testing.	Validation is Dynamic Testing.
3	Quality assurance comes under verification.	Quality control comes under validation.
4	In verification, the execution of the code does not happen.	In Validation, the execution of the code happens.
5	In verification, it is verified whether the inputs follow the outputs or not.	In Validation, it is validated whether the user accepts the product or not.
6	Verification is done before validation testing.	Validation testing takes place after verification testing.
7	Here it is checked whether we are developing the right product or not.	Here it is checked whether the product developed is right or not.

42. What is the difference between Bug, Defect, Error, Fault, and Failure?

1. **Bug-** It is a flaw in the software which means that the software is not working as per the requirement. When there is a coding error, it leads to a program breakdown, which is known as a bug.
2. **Defect-** It occurs when an application is not working as per the requirements. It is the deviation or the difference between the expected output and the actual output.

3. **Error-** It is a mistake made in the code due to which the code cannot be executed.
4. **Fault-** It can be termed as a condition that causes the software to fail to perform its required function according to the specification.
5. **Failure-** It is the inability of the software or system to perform the required function due to the accumulation of several defects that ultimately results in the loss of information in critical modules and thus make the system unresponsive.

43. What do you verify in white-box testing?

The main aim of white-box testing is to verify the following areas in the software-

- Security loopholes in the source code.
- Conditions of all the loops and overall functionality of the software.
- Expected output.
- Poorly structured paths in the coding processes.
- Line-by-line verification of the code.
- The flow of the software structure is mentioned in the software requirement document.

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

S No.	Functional Testing	Non-Functional Testing
1	In functional testing, the behavior of the application is validated.	In non-functional testing, the performance of the application is validated.
2	It is based on customers' requirements.	It is based on customers' expectations.S No. Functional Testing
3	It describes what the product does.	It describes how the product works.
4	Functional testing is performed before non-functional testing.	Non-functional testing is performed after functional testing.
5	It is more convenient to conduct functional testing by manual testing.	It is very hard to perform non-functional testing manually.
6	Some types of functional testing are- • Unit testing • Smoke testing • Integration testing • Regression testing	Some types of Non-functional testing are- • Performance testing. • Volume testing • Scalability • Usability testing • Load testing

45. What is the difference between data-driven testing and retesting?

S No.	Data-driven testing	Retesting
1	It is an automated testing procedure.	It is a manual testing procedure.
2	In this, the application is tested with multiple test data.	In this, the application is tested with an entirely new set of data.
3	Most of the time this testing is part of regression testing.	Most of the time this testing is independent of regression testing.
4	It is a very easy procedure than retesting.	It is a very tedious and boring procedure as the tester needs to give input manually.

46. Why should developers not test the software that they build?

Some of the reasons why developers should not test their own software are-

- **Unconscious bias-** Someone testing a product that they created may raise unintentional bias in the testing process. Software developers lack the objectivity to be able to test their own work. A dedicated tester is a neutral party that can see things in an impartial way and avoids unintentional bias.
- **Regression confusion-** The requirements are being considered by a developer as only functionalities and if any of the requirements are misunderstood by the user then there will be a failure in the application. tester on the other hand views requirements from the business point of view so they can accomplish what is required.
- **Weak end-to-end perspective-** Developers tend to focus on a single task or functionality that is chosen for the day and they work with a single focus but testers, on the other hand, think broadly about the functionality along with the full perspective of the application. This helps to complete the application earlier.
- **Less experience-** The software tester has vast experience in testing various applications and they are aware of the common bugs and tough application logic. Due to a rich knowledge base, the tester is able to find bugs easily and test them whereas the developers have a skill set only for fixing the broken application but not for breaking an application and finding bugs.
- **Less time-** For testing, it requires hard work, focus, and most of all dedicated time. If the developer is responsible for testing also, then apart from writing the code which is also a time-consuming process, they have to create strategies and plans, write project documentation, and carry out unit tests. It’s not realistic to assume that they will do their job with so many roles and responsibilities with 100% capacity. For maximum efficiency, and to assure that no quality is lost, coding and testing are two different roles for two different skillset persons.
- **Slows down release time-** If the developer is responsible for both writing code and conducting testing, then both jobs cannot be done simultaneously. Coding

has to stop for testing to begin and vice versa. This will end up slowing productivity and will eventually affect the release time of the software.

- **QA testers are specially trained-** It may be possible that the developer has a skill set required for QA testing the product but this doesn't mean that they are the best person for this job. QA testers have experience, background, and technical perspectives that are diverse and different from developers.

47. What qualities should a software developer possess?

The top seven qualities of the software developer are-

- **Technical expertise-** A developer must have strong technical expertise so that he/she knows how to achieve an optimal solution for a particular task. Their code will not be clear and understandable but also follow the coding standards.
- **Maintains an end-user focus-** An exceptional developer will want to first fully understand the business case for the major projects they undertake. This is to ensure that their actions will not only affect the end-users but ultimately the organization.
- **Ability to learn, grow, and adapt-** Continuous improvement is a good practice for both the software and the engineer.
- **Time management-** A software developer must be a master at managing their time in order to cram so much action into every single working day.
- **Interpersonal skills-** Some of the key interpersonal skills that a software engineer needs are Collaboration skills, Communication skills, Empathy skills, Critical thinking skills, and Leadership skills.
- **Communication skills-** A good developer must know how to communicate, clarify, explain, and persuade.
- **Curiosity-** Good developers are inquisitive. They tend to ask themselves and their peer a lot of questions while they work.
- **Good team player-** A good developer shares their knowledge and believes in upscaling their team. They offer teammates help when they face problems that they cannot solve.

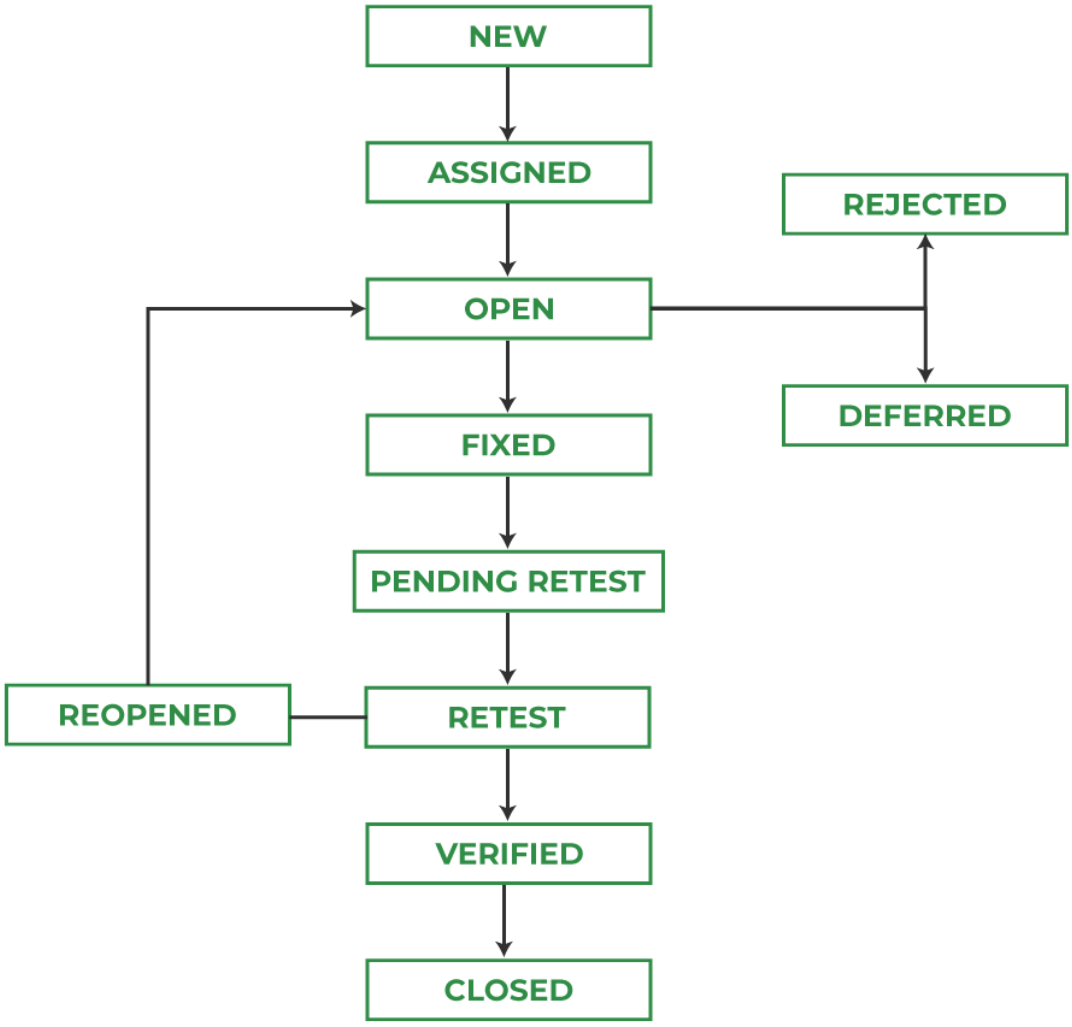
48. What are the benefits of acceptance testing?

Acceptance testing is based on user requirements and function processing. It is also known as User Acceptance Testing (UAT). It is a process that verifies if a solution is conforming to specified requirements and user requirements or not. It is done by the customer before accepting the final product. Some of the benefits of user acceptance testing are-

- It increases the satisfaction of the customer as they test the application itself.
- Reduces the risk of defects being identified in the production.
- This will help end-users to gain skills and confidence while using the new system prior to going live.
- It improves the requirement definition document as the client tests the requirement definition according to his needs.
- The information gathered through acceptance testing is used by the stakeholders to better understand the requirements of the targeted audience.

49. Explain the bug life cycle.

The bug life cycle in software testing is the specific set of states that a bug goes through in its entire life. The purpose here is to easily communicate the current status of defects and make the bug-fixing process easy and efficient. Below lifecycle diagram covers all possible states of the bug lifecycle-



- **New-** When a new defect is detected and posted for the first time then it is assigned a status as “New”.
- **Assigned-** Once the bug is posted by the tester, it is approved by the lead of the tester, and the bug is assigned to the developer team. The status is changed to “Assigned”.
- **Open-** The developer starts analyzing the bug and works on fixing the bug. The status is changed to “Open”.
- **Fixed-** When the developer makes changes in the code to fix the bug and verifies the changes, he or she can make the status of the bug “Fixed”.
- **Pending Retest-** Once the defect is fixed, the developer gives the particular code to the tester to retest the code.
- **Retest-** The tester retests the code to check whether the defect is fixed or not and changes the status to “Retest”.
- **Reopen-** The tester changes the status to “Reopen” if the bug persists even after the developer fixed the bug.
- **Verified-** The tester retests the bug after it is fixed by the developer. If there is no bug in the software then the status is changed to “Verified”.
- **Closed-** If the bug no longer exists, then the tester assigns the status “Closed”.
- **Rejected-** If the developer feels that the defect is not a genuine defect then it changes the defect state to “Rejected”.

- **Deferred-** The status of the defect is changed to “Deferred” **i**f the present bug is not of prime priority and if it is expected to get fixed in the next release.
- **Duplicate-** If the bug is repeated twice, the status is changed to “Duplicate”.
- **Not a defect-** If the detected bug or defect does not affect the functionality of the application then the status is changed to “Not a defect”.
- **Can’t be fixed-** If it is not possible to fix the bug due to one of the following reasons technology not supporting, the cost of fixing is more, then the status is changed to “Can’t be Fixed”.
- **Need more information-** If the developer is unable to reproduce the defect as per the steps provided then the status is changed to “Need More Information”. In this case, the tester needs to add detailed steps to reproduce the bug and assign the bug back to the development team for fixing it.
- **Not reproducible-** If it is not possible for the developer to reproduce the bug due to one of the following reasons platform mismatch, improper defect document, data mismatch, build mismatch, or inconsistent defect. Then the status is changed to “Not reproducible”.

50. What is the role of Usability Testing?

Usability testing means determining the ease with which an end-user can easily access the application with or without programming language knowledge. It is also known as User Experience testing which is recommended during the initial design phase of SDLC. It is done to serve the following purpose-

- To identify the usability errors in the system early in the development cycle.
- It can help to save products from failure.
- It minimizes the risk of product failure.
- Usability testing increases the likelihood of usage and repeat usage.