Module 2 (Manual Testing)

1. What is Exploratory Testing?

Ans.

Exploratory testing is type of software testing in which the tester is free to select any possible methodology to test the software. It is an unscripted approach for software testing. In exploratory testing, software developers use their personal learning, knowledge, skills, and abilities to test the software developed by themselves. Exploratory testing checks the functionality and operations of the software as well as it identifies the functional and technical faults in it.

2. What is traceability matrix?

Ans. Requirements tracing, a process of documenting the links between the requirements and the work products developed to implement and verify those requirements. The RTM captures all requirements and their traceability in a single document delivered at the conclusion of the life cycle.

RTM - Workflow: The Matrix is created at the very beginning of a project as it forms the basis of the project's scope and deliverables that will be produced.

The Matrix is bi-directional, as it tracks the requirement forward by examining the output of the deliverables and backward by looking at

the business requirement that was specified for a particular feature of the product.

Requirement traceability Matrix - Parameters:

- Requirement ID
- Risks
- Requirement Type
- Requirement Description
- Trace to Design Specification
- Unit Test Cases
- Integration Test Cases
- System Test Cases
- User Acceptance Test Cases
- Trace to Test Script

Requirement traceability Matrix - Parameters:

- Requirement ID
- Risks
- Requirement Type
- Requirement Description
- Trace to Design Specification
- Unit Test Cases
- Integration Test Cases
- System Test Cases

• User Acceptance Test Cases

• Trace to Test Script

3. What is Boundary value testing?

Ans. Boundary value analysis is a black box software testing technique where test cases are designed using boundary values. BVA is based on the single fault assumption, also known as critical fault assumption which states that failures are rarely the product of two or more simultaneous faults. Hencewhile designing the test cases for BVA we keep all but one variable to take the extreme value.

Test case design for BVA:

while designing the test cases for BVA first we determine the number of inputs variables in the problem. For each input variable, we then determine the range of values it can take. Then we determine the extreme values and nominal value for each input variable.

4. What is Equivalence partitioning testing?

Ans.

Equivalence partitioning is also known as equivalence class partitioning (ECP). It is a software testing technique or black box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

5. What is Integration testing?

Ans.

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of

integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

Integration Strategies:

- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration
- Hybrid Integration

6. What determines the level of risk?

Ans.

Risk can be defined as the probability of an event, hazard, accident, threat or situation occurring and its undesirable consequences. It is a factor that could result in negative consequences and usually expressed as the product of impact and likelihood.

In software terminology, the risk is broadly divided into two main categories:

Project Risks:

- Supplier issues
- Organizational factors
- Technical issues **Product Risks:**
- Below are some of the product risks occurring in a LIVE environment?
- Defect Prone Software delivered
- The Critical defects in the product that could cause harm to an

individual (injury or death) or company

• Poor software Features

• Inconsistent Software Features

7. What is Alpha testing?

Ans. Alpha

testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha testing is one of the user acceptance testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software.

8. What is beta testing?

Ans.

Beta testing is performed by real users of the software application in a real environment. Beta testing is one of the types of user acceptance testing. A beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality.

9. What is component testing?

Ans.

Also known as unit testing, module testing or program testing. A unit is the smallest testable part of software. Unit testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed. Unit tests are typically written andrun by software developers to ensure that code meets its design and behaves as intended with a debugging tool.

10. What is functional system testing?

Ans.

Functional testing is a type of software testing in which the system is tested against the functional requirements and specification.

Functional testing ensures that the requirements or specification are properly satisfied by the application.

This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions.

It is basically defined as a type of testing which verifies that each function of the software.

application works in conformance with the requirement and specification.

It focuses on simulation of actual system usage but does

not develop any system structure assumptions.

11.What is Non-Functional Testing?

Ans.

Nonfunctional testing is a type of software testing that is performed to verify the non-functional requirements of the application. It verifies whether the behaviour of the system is as per the requirement or not. It tests all the aspects which are not tested in functional testing. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing. Non-functional testing is as important as functional testing.

12. What is GUI Testing?

Ans.

Graphical User Interface Testing (GUI) is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application. GUI testing generally evaluates a design of elements such as layout, colour and also fonts, font size, labels, textboxes, text formatting, captions, buttons, lists, icons, links, and content. GUI testing processes may be either manual or automatic and are often performed by thirdparty companies, rather than developers or and users.

13. What is Adhoc testing?

Ans.

Adhoc testing is a type of software testing which is performed informally and randomly after than formal testing is completed to find out any loophole in the system. For this reason, it is also known asrandom testing or monkey testing. Adhoc testing is not performed in

an structured way so it is not based on any methodological approach. That's why adhoc testing is a type of unstructured software testing.

14. What is load testing?

Ans.

• Load testing - Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

• Load testing is a kind of performance testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

• **This testing sually identifies–** The maximum operating capacity of an application

• Determine whether current infrastructure is sufficient to run the application

• Sustainability of application with respect to peak user load

• Number of concurrent users that an application can support, and scalability to allow more users to access it.

15. What is stress Testing?

Ans.

Stress testing - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous inputto system or database load.

• Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.

• It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.

• Stress Testing is done to make sure that the system would not crash under crunch situations.

• Stress testing is also known as endurance testing.

• Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity.

16. What is white box testing and list the types of white box testing?

Ans.

Testing based on an analysis of the internal structure of the component or system. Structure-based testing technique is also known as white boxor glass box testing technique because here the testers require knowledge of how the software is implemented, how it works.Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.

White box testing is the detailed investigation of internal logic and structure of the code.

Test/code coverage

Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and this is known as test coverage.

$$\text{Coverage} = \frac{\text{Number of coverage item exercised}}{\text{Total number of coverage items}} \times 100\%$$

Types of coverage

Statement/segment coverage:

The statement coverage is also known as line coverage or segment coverage.

$$\text{Statement coverage} = \frac{\text{Number of statement exercised}}{\text{Total number of statements}} \times 100\%$$

Decision/Branch coverage:

Decision coverage also known as branch coverage or all – edges coverage.

Decision coverage = Number of decision outcomes exercised

$$\frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$

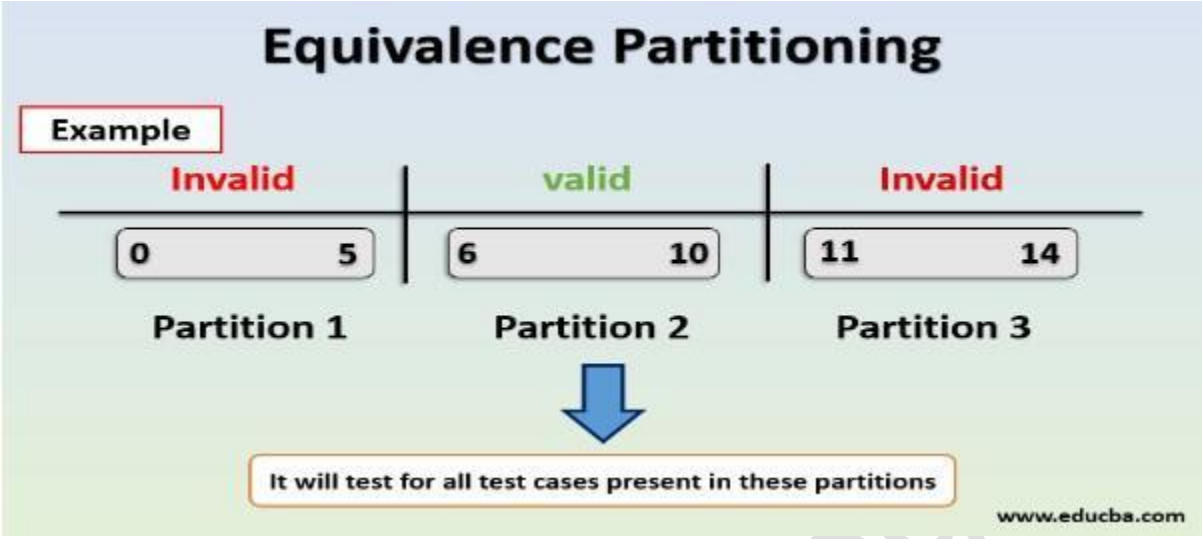
17. What is black box testing? What are the different black box testing techniques?

Ans.

Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Black box testing techniques:

1. **Equivalence partitioning:** - Equivalence partitioning is also known as equivalence class partitioning (ECP). It is a software testing technique or black box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

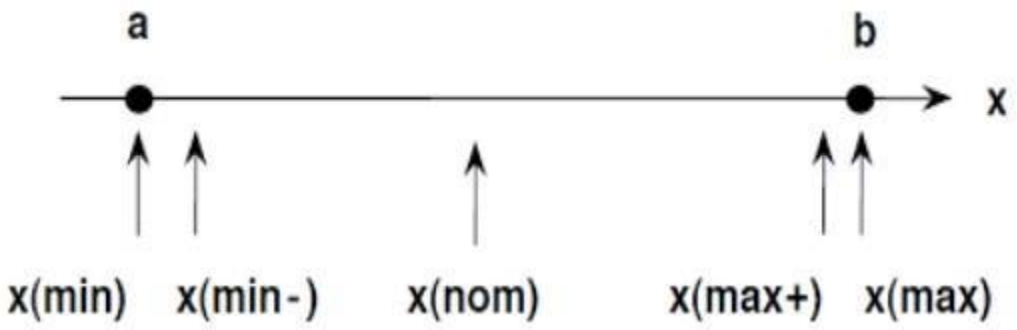


2. Boundary value analysis:-

classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed. In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, equivalence class represents or describes set of valid or invalid states.

Boundary value analysis is a black box software testing technique where test cases are designed using boundary values. BVA is based on the single fault assumption, also known as critical fault assumption which states that failures are rarely the product of two or more simultaneous faults. Hence while designing the test cases for BVA we keep all but one variable to take

the extreme value. Test case design for BVA:



3. **Decision table:** - Decision table are used in various engineering fields to represent complex logical relationship. This testing is a very effective tool in testing the software and its requirements management.

Parts of decision table : In software testing the decision table has 4 parts which are divided into portions and are given below:

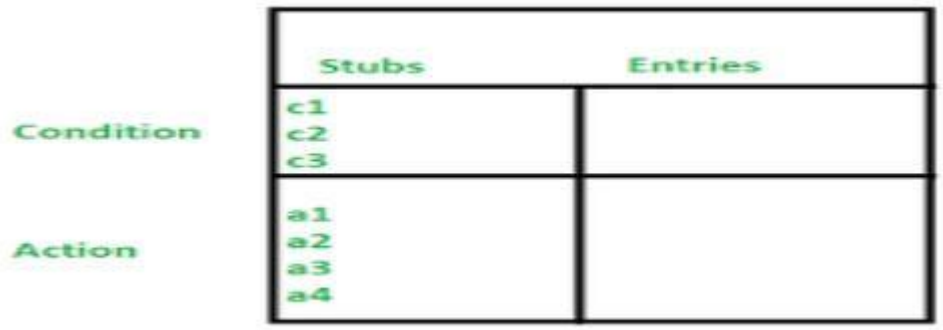
1. Condition stubs: - The conditions are listed in this first upper left part of the decision table that is used to determine a particular action or set of actions.

2. Action stubs: - All the possible actions are given in the first lower left portion (i.e., below condition stub) of the decision table.

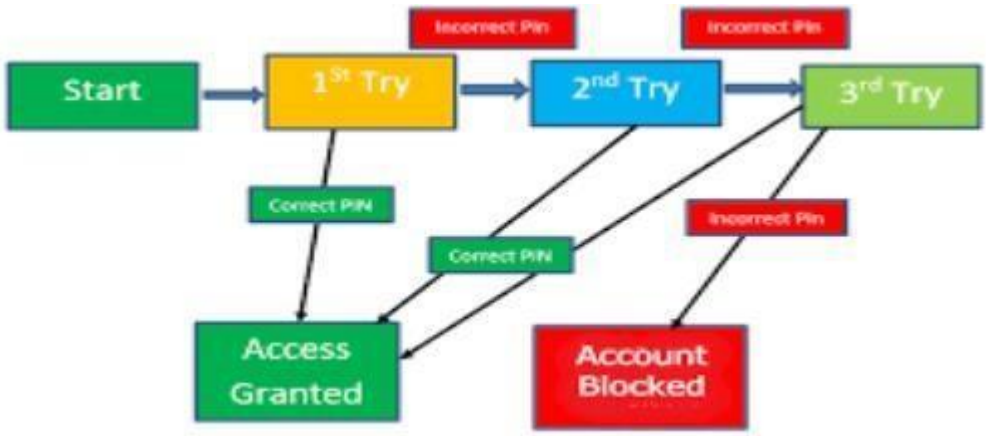
3. Condition entries: - In the condition entry, the values are inputted

in the upper right portion of the decision table. In the condition entries part of the table, there are multiple rows and columns which are known as rule.

4. Action entries: - In the action entry, every entry has some associated action or set of actions in the lower right portion of the decision table and these values are called outputs.



4. State transition testing: - State transition testing is a type of software testing which is performed to check the change in the state of the application under varying input.



5. Use case testing: - A use case is as a tool for defining the required user interaction and if you are trying to create a new application ar make changes to an existing application, several discussions are made. Use case testing is generally a part of a black box testing and that helps developers and testers to identify test scenarios that exercise that whole system on each transaction basis from start to finish.

Feature of use case testing: -

There is some feature of a use case testing,
which is used to test the software project and provide s better response, these are given below:

1.

Use case testing is not testing that is performed to decide the

quality of the software.

2.

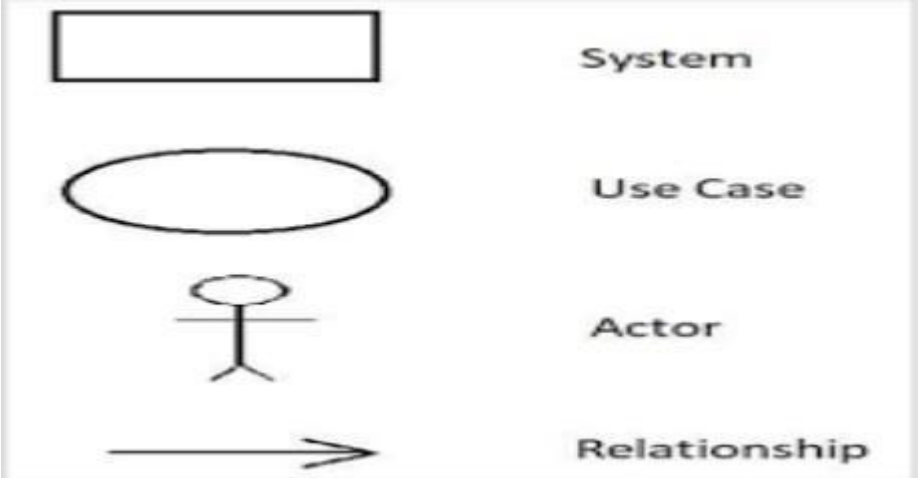
Although it is a type of end to end testing, it won't ensure the entire coverage of the user application.

3.

Use case has generally captured the interactions between 'actors' and the 'system'.

4.'Actors' represents the user and their interactions that each user takes part in.

5. The use case will find out the defects in integration testing.



18. Mention what are the categories of defects?

Ans.

Data Quality/Database Defects: Deals with improper handling of data inthe database.

Examples:

• Values not deleted/inserted into the database properly

• Improper/wrong/null values inserted in place of the actual values

Critical Functionality Defects: The occurrence of these bugs hampers the

crucial functionality of the application.

Examples: - •Exceptions

Functionality Defects: These defects affect the functionality of the

application.

Security Defects: Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.

Examples:

- Improper error/warning/UI messages
- Spelling mistakes
- Alignment problems

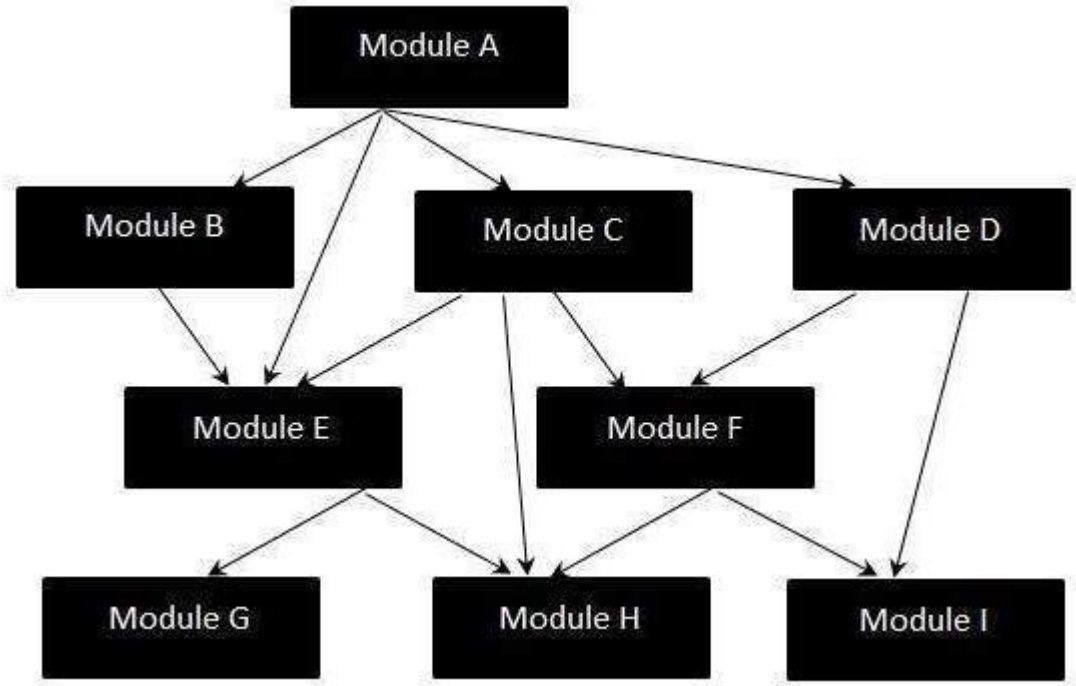
19. Mention what bigbang testing is?

Ans.

Big

Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

Big Bang Testing is represented by the following workflow diagram:



19. What is the purpose of exit criteria?

Ans.

Exit

criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

Examples of Exit Criteria:

- Verify if all tests planned have been run.
- Verify if the level of requirement coverage has been met.

• Verify if there are NO Critical or high severity defects that are left outstanding.

20. When should "Regression Testing" be performed?

• **Ans.** Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

• You also need to ensure that the modifications have not caused unintended side – effects elsewhere and that the modified system still meets its requirements.

Regression testing should be carried out:

• When the system is stable and the system or the environment.

• Changes when testing bug – fix releases as part of the maintenance phase.

• It should be considered complete when agreed completion criteria for regression testing have been met.

• Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation.

21. What is 7 key principles? Explain in detail?

Ans.

Software testing is the process of executing a program with the aim of finding the

error. To make our software perform well it should be error- free. If

testing is done successfully it will remove all the errors from the software.

There are seven principles in software testing:

1. Testing shows the presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defects clustering
5. Pesticide paradox
6. Testing is context-dependent
7. Absence of errors fallacy

1. Testing shows the presence of defects:

The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about thepresence of defects and doesn't talk about the absence of defects.

2.Exhaustive testing is not possible:

It is process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-condition is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case.

3. Early testing: To find the defect in the software, early test activity shall be started. The defect detected in the early phase of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

4. Defects clustering: In a project, a small number of modules can contain most of the defects. Pareto principle to software testing state that 80% of software defect comes from 20% of modules.

5. Pesticide paradox: Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

6. Testing is context-dependent: The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, The testing of the

ecommerce site is different from the testing of the android application.

7. Absence of errors fallacy: If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil all the customer requirements.

22. Difference between QA v/s QC v/s Tester

Ans.

Quality Assurance(QA)	Quality Control(QC)	Tester
• It focuses on providing assurance that the quality requested will be achieved.	• It focuses on fulfilling the quality requested.	• It is responsible for evaluating individual software.
• It is the technique of	• It is the technique to	• It simply evaluates functionality of

managing quality.	verify quality.	software application.
• It is involved during the development phase.	• It is not included during the development phase.	• Software tester generally test whether or not code runs as we expected it to run.
• It does not include the	• It always includes the	• They are responsible for quality of software
execution of the program.	execution the program.	development and deployment.
• It is a preventive technique.	• It is a corrective technique.	• Tester should have deep knowledge of system that is being developed, good communication

		skills, critical thinking, etc
It pays main focus is on the intermediate process.	• Its primary focus is on final products.	
• All team members of the project are involved.	• Generally, the testing team of the project is involved.	

23. Difference between Smoke and Sanity?

Ans.

Smoke Testing	Sanity Testing
• Smoke testing is done to assure that the acutefunctionalities of program is working fine.	• Sanity testing is done to check the bugs have been fixed after the build.

• Smoke testing is also calledsubsetof acceptance testing.	• Sanity testing is also called subset of regression testing
• Smoke testing is documented.	• Sanity testing isn't documented.
• Smoke testing is performed by either developers or testers.	• Sanity testing is normally performed by testers.
• Smoke testing may be stable or unstable.	• Sanity testing is stable.
• Smoke testing is scripted.	• Sanity testing is usually not scripted.
• Smoke testing is done to measures the stability of the system/product by performing testing.	• Sanity testing is done to measures the rationality of the system/product by performing testing.

• Smoke testing is used to test all over function of the system/product.	• Sanity testing is used in the case of only modified or defect function of system/products.

24. Difference between verification and Validation.

Ans.

Verification	Validation
• It includes checking documents, design, codes and programs.	• It includes testing and validating the actual product.
• Verification is the static testing.	• Validation is the dynamic testing.
• It does not include the execution of the code.	• It includes the execution of the code.

• Methods used in verification are reviews, walkthroughs, inspections and deskchecking.	• Methods used in validation are black box testing, white box testing and nonfunctional testing.
• It checks whether the software conforms to specification or not.	• It checks whether the software conforms to specification or not.
• It can find the bugs in the early stage of the development.	• It can only find the bugs that could not be found by the verification process.
• The goal of verification is application and software architecture and specification.	• The goal of validation is an actual product.

25. Explain types of Performance testing.

Ans.

Performance

Testing is a type of software testing that ensures software applications to perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity and stability under a particular workload.

Performance Testing Attributes :

- **Speed:**

It determines whether the software product responds rapidly.

- **Scalability:**

It determines amount of load the software product can handle at a time.

- **Stability:**

It determines whether the software product is stable in case of varying workloads.

- **Reliability:**

It determines whether the software product is secure or not.

Objective of Performance Testing:

1. The objective of performance testing is to eliminate performance congestion.
2. It uncovers what is needed to be improved before the product is launched in market.
3. The objective of performance testing is to make software rapid.
4. The objective of performance testing is to make software stable and reliable.

Types of Performance Testing:

1. **Load testing:**

It checks the product's ability to perform under anticipated user loads. The objective is to identify performance congestion before the software product is launched in market.

2. **Stress testing:**

It involves testing a product under extreme workloads to see whether it handles high traffic or not. The objective is to identify the breaking point of a software product.

3. **Endurance testing:**

It is performed to ensure the software can handle the expected load over a long period of time.

4. **Spike testing:** It tests the product's reaction to sudden large spikes in the load generated by users.

5. **Volume testing:**

In volume testing large number of data is saved in a database and the overall software system's behaviour is observed. The objective is to check product's performance under varying database volumes.

6. **Scalability testing:**

In scalability testing, software application's effectiveness is determined in scaling up to support an increase in user load. It helps in planning capacity addition to your software system.

26. Explain the difference between Functional testing and NonFunctional testing.

Ans.

Functional testing	Non-Functional testing
• It verifies the operations and actions of an application.	• It verifies the behaviour of an application.
• It is based on requirements of customer.	• It is based on expectations of customer.
• It helps to enhance the behaviour of the application.	• It helps to improve the performance of the

	application.
• Functional testing is easy to execute manually.	• It is hard to execute nonfunctional testing manually.
• It tests what the product does.	• It describes how the product does.
• Functional testing is based on the business requirement.	• Non-Functional testing is based on the performance requirement.
• Example: 1.Unit testing 2.Smoke testing 3.Integration testing 4.Regression testing	• Example: 1.performance testing 2.Load testing 3.Stress testing 4.Scalability testing

26. What is the difference between the STLC (Software Testing Life Cycle) and SDLC(Software Development Life Cycle)?

Ans.

SDLC	STLC
• SDLC is mainly related to software development.	• STLC is mainly related to software testing.
• Besides development other phases like testing is also included.	• It focuses only on testing the software.
• SDLC involves total six phases or steps.	• STLC involves only five phases or steps.
• In SDLC, more number of members (developers) are required for the whole process.	• In STLC, less number of members (tester) are needed.
• In SDLC, more number of members (developers) are required for the whole process.	• In STLC, less number of members (tester) are needed.

• In SDLC, development team make the plans and designs based on the requirements.	• In SDLC, development team make the plans and designs based on the requirements.
• Goal of SDLC is to complete successful development of software.	• Goal of STLC is to complete successful testing of software.
• It helps in developing good quality software.	• It helps making the software defects free.
• SDLC phases are completed before the STLC phases.	• STLC phases are performed after SDLC phases.

27. What is the difference between test scenarios, test cases, and test script?

Ans.

Test scenario	Test cases	Test script

• The test scenario is just a document that is detailed and provides details about the assessment method, testing process, precondition, and anticipated output.	• Test cases is a step by step procedure to test any functionality of the software application/product.	• Test script is set of instruction or a short program to test any functionality of software application/product.
• The test scenarios are the ones based on the use situation and give one-line information one what to check.	• Test cases is a manual approach of software testing.	• Test script is an automatic approach of software testing.

• Test scenarios are one-liner statement, however, it is linked to a few test. • These are highlevel actions.	• It is a set up that is used by the tester to test any specific function of the softwareproduct • Point by point test case configuration encourages tester to test viably.	• It is a program developed by the tester, intended to test any specific function. • Automatic testing approach is beneficial for constant execution.
• Writing the test scenario's primary objective is an address end to get rid of functionality of a software	• Test cases are written by Manually.	• Test scripting is done by scripting format.

program.		
• It will take less time as compared to test cases.	• Test case is developed in form of templates.	• Test script is developed in form of scripting.
• The test scenario will help us in a way that is nimble of through the functionality.	• If the tester does not have a good understanding of how the program is used or about the recent risks to the program, then it will be difficult to use the test cases properly.	• Active software projects frequently change.so testers have to make a continuous effort to update the scripts to match the changes of the new product.
• Test scenario are really easy to	• Test case is used in manual testing	• Test script is used in automatic

maintain due to their highlevel design.	environment.	testing environment.
• The test scenarios tend to be work on the essential to “things to be tested”.	• Test cases are classified as delegated, positive, reusable, negative and UI test cases.	• Test script are characterized as manual test script and automatic test scripts.
• Requires fewer resources and less time.	• Requires more resources and time.	• Requires less time for testing scripts.

28. Explain what Test Plan is? What is the information that should be covered.

Ans.

• A test plan is a detailed document that describes the test strategy,

objectives, schedule, estimation, deliverable, and resources required to perform testing for a software product.

• As per ISTQB definition: “test plan is a document describing the scope, approach, resources, and schedule of intended test activities.”

1. Introduction to the test plan document
2. Assumptions when testing application
3. List of test cases included in testing the application
4. List of features to be tested
5. What sort of approach to use when testing the software
6. List of deliverables that need to be tested
7. They resources allocated for testing the application
8. Any risks involved during the testing process
9. A schedule of tasks and milestones as testing is stared

29. What are the different Methodologies in Agile Development Model?

Ans.

There are various methodologies present in agile testing and those are listed below: Scrum eXtreme programming.

Below listed methodologies are used less frequently.

• **Dynamic system development method (DSDM):** This is an iterative and incremental approach that emphasizes on the continuous user involvement.

• **Total driven development (TDD):** This is a technique which has short iterations where new test cases covering the desired improvement or new functionality are written first.

• **Feature driven development:** This is an iterative and incremental software development process and this can aim depends on the features.

30. Explain the difference between Authorization and Authentication in Web testing.What are the common problems faced in Web testing.?

Ans.

Authentication	Authorization
In the authentication process, the identity of users are checked for providing the access to the system.	While in authorization process, a the person's or user's authorities are checked for accessing the Resources.

In the authentication process, users or persons are verified.	While in this process, users or persons are validated.
It is done before the authorization process.	While this process is done after the authentication process.
It needs usually the user's login details.	While it needs the user's privilege or security levels.
Authentication determines whether the person is user or not.	While it determines What permission does the user have?
Example: Employees in a company are required to authenticate through the network before accessing their company email.	Example: After an employee successfully authenticates, the system determines what information the employees are allowed to access.
The user authentication is visible at user end.	The user authorization is not visible at the user end.
The user authentication is	The user authorization is carried

identified with username, password, face recognition, retina scan, fingerprints, etc.	out through the access rights to resources by using roles that have been pre-defined.