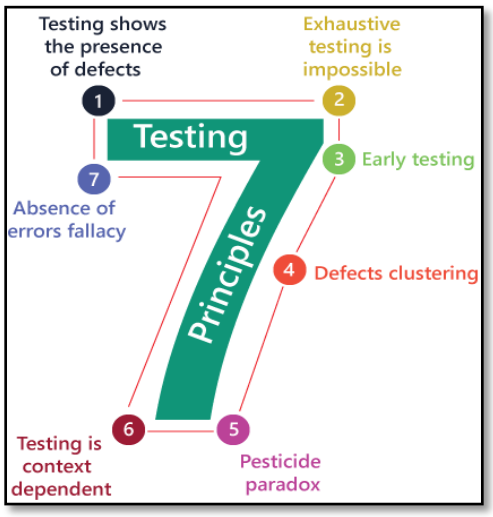
Software Testing Assignment

:: Module 2 : Manual Testing ::

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

Ans. For testing a software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.



1. Testing shows presence of Defects:

- Testing can show that defects are present, but cannot prove that there are no defects.
- Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
- We test to find Faults

2. Exhaustive Testing is Impossible!

- Testing everything including all combinations of inputs and preconditions is not possible.
- So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts
- Why do not Testing Everything?
    - Exhaustive testing of complex software applications:
        - requires enormous resources
        - is too expensive
        - takes too long
- It is therefore impractical.

3. Early Testing:

- Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
- Testing activities should start as early as possible in the development life cycle.
- software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

4. Defect Clustering:

- A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.
- Defects are not evenly spread in a system.
- In other words, most defects found during testing are usually confined to a small number of modules.
- software testing states that 80% of software defects come from 20% of modules.

5. The Pesticide Paradox:

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
- To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
- Testing identifies bugs, and programmers respond to fix them As bugs are eliminated by the programmers, the software improves As software improves the effectiveness of previous tests erodes.

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 e-commerce site is different from the testing of the Android application.
- Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing
- 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software.
- 1 to 3 failures per KLOC typical for industrial software 0.01 failures per KLOC for NASA Shuttle code!
- Also different industries impose different testing standards.

7. Absence of Errors Fallacy:

- Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations.
- If a built software is 99% bug-free but 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 fulfill all the customer requirements.

2. What is Error, Defect, Bug and failure?

Ans. A mistake in coding is called error.

- Error found by tester is called defect.
- Defect accepted by development team then it is called bug.
- Build does not meet the requirements then it is failure.

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

Ans.

No.	Quality Assurance	Quality Control	Testing
1.	It can be described as the planned activities that are carried out in a systematic manner to ensure fulfilment of requirements.	QC is a method of identify bugs, defects and other quality related issues, along with its study, analysis and correction.	It is a process of identifying and locating the bugs, defects and other vulnerable issues, in a software product.
2.	QA may be seen as a subset of a testing life cycle.	It is a subset of quality assurance.	A subset of quality control process.
3.	It emphasizes on processes and procedures rather than carrying out actual testing on the product.	It focuses on quality attribute of a product, by adopting and implementing various activities, including testing.	Testing is one of the techniques of the quality control, which identify bugs and defects in a product, so as to improve its quality.
4.	It is about managing the quality.	It concerns with quality verification.	It concerns with the improvement of the quality.
5.	QA is planning.	QC involves the execution of specific and certain activities and methods, in the direction of achieving quality.	Testing involves the execution of the tests, to evaluate a software product or its functionalities.
6.	QA defines standards and methodologies which lays the foundation for achieving user requirements.	QC is to make sure that the standards are followed in order to implement the work product.	It ensures the identification and detection of the bugs/ defects (if any) in a software product.
7.	QA is responsible for the entire Software Development Life Cycle.	QC is responsible for software testing life cycle.	It is accountable for the bug/defect free product.
8.	It is a process oriented approach.	A product oriented approach.	Testing is also a product-Oriented approach.
9.	QA is a failure prevention system.	QA is a corrective measure mechanism	Testing is a failure detection system.

4. Difference between verification and Validation.

	Verification	Validation
Definition	Verification refers to the set of activities that ensure software correctly implements the specific function	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
Focus	It includes checking documents, designs, codes, and programs.	It includes testing and validating the actual product.
Type of Testing	Verification is the static testing.	Validation is dynamic testing.
Execution	It does not include the execution of the code.	It includes the execution of the code.
Methods Used	Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
Purpose	It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
Bug	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.
Goal	The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Responsibility	Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
Timing	It comes before validation.	It comes after verification.
Human or Computer	It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.
Lifecycle	After a valid and complete specification the verification starts.	Validation begins as soon as project starts.
Error Focus	Verification is for prevention of errors.	Validation is for detection of errors.
Another Terminology	Verification is also termed as white box testing or static testing as work product goes through reviews.	Validation can be termed as black box testing or dynamic testing as work product is executed.
Performance	Verification finds about 50 to 60% of the defects.	Validation finds about 20 to 30% of the defects.
Stability	Verification is based on the opinion of reviewer and may change from person to person.	Validation is based on the fact and is often stable.

5. Explain the difference between Functional testing and Non Functional testing.

Ans.

No.	Functional Testing	Non Functional Testing
1	Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non Functional Testing checks the Performance, reliability, scalability and other aspects of the software system.
2	Functional testing is executed first	Non functional testing should be performed after functional testing
3	Manual testing or automation tools can be used for functional testing	Using tools will be effective for this testing
4	Business requirements are the inputs to functional testing	Performance parameters like speed, scalability are inputs to non-functional testing.
5	Functional testing describes what the product does	Nonfunctional testing describes how good the product works
6	Easy to do manual testing	Tough to do manual testing
7	It verifies the operations and actions of an application.	It verifies the behaviour of an application.
8	It is based on requirements of customer.	It is based on expectations of customer.
9	It helps to enhance the behaviour of the application.	It helps to improve the performance of the application.
10	Functional testing is based on the business requirement.	Non-functional testing is based on the performance requirement.
11	Types of Functional testing are · Smoke Testing · Sanity Testing · White box testing · Black Box testing · End to end testing · Experienced based testing	Types of Nonfunctional testing are · Performance Testing · Load Testing · Volume Testing · Stress Testing · Security Testing · Installation Testing · Penetration Testing · Compatibility Testing · Migration Testing

6. What is Boundary value testing? (BVA- Boundary Value Analysis)

Ans. Boundary value testing is a black-box testing technique that involves testing the boundaries of a system or application to ensure that it behaves correctly at the extremes of its expected input or output values.

The idea is to test the system at the edges of its expected operating range, where errors are more likely to occur. This includes testing:
- Minimum and maximum values
- Just inside and just outside the expected range
- Empty or null values
- Special values like 0, infinity, or NaN (not a number)

By testing these boundary values, you can identify errors and bugs that might not be caught through normal testing.

Some examples of boundary value testing include:
- Testing a password field with the minimum and maximum allowed length
- Testing a numeric field with the minimum and maximum allowed value
- Testing a date field with the earliest and latest allowed date

Boundary value testing is an important part of ensuring the reliability and robustness of software systems.

7. What is Equivalence partitioning testing? (EP) (ECP)

Ans. **Equivalence Partitioning or Equivalence Class Partitioning** is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc.In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

- It divides the input data of software into different equivalence data classes.
- You can apply this technique, where there is a range in the input field.

Example 1: Equivalence and Boundary Value

- Let's consider the behaviour of Order Pizza Text Box Below
- Pizza values 1 to 10 is considered valid. A success message is shown.
- While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered"

Order Pizza:
Here is the test condition

1. Any Number greater than 10 entered in the Order Pizza field (let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.

We cannot test all the possible values because if done, the number of test cases will be more than 100. To address this problem, we use equivalence partitioning hypothesis where we divide the possible values of tickets into groups or sets as shown below where the system behaviour can be considered the same.

8. Mention what big bang testing is?

Ans. Big Bang testing is a software testing approach where the entire application is tested all at once, rather than breaking it down into smaller components or modules. This approach involves testing the complete system, with all its components and integrations, in a single testing phase.

In Big Bang testing, testers typically verify that the entire system works as expected, without focusing on individual parts or workflows. This approach can be useful for small projects or when there are significant changes to the entire system. However, it may not be suitable for large, complex systems, as it can be difficult to isolate and identify specific issues.

Some pros of Big Bang testing include:

- Time-efficient, as it tests the entire system at once
- Can be useful for small projects or significant system changes

Cons include:

- Difficult to isolate and identify specific issues
- May not be suitable for large, complex systems
- Can be challenging to manage and execute

9. What is the purpose of exit criteria?

Ans. Exit criteria are a set of predetermined conditions or requirements that must be met before a project, process, or phase can be considered complete and ready to move on to the next stage or be closed.

Purpose of exit criteria is to define when we STOP testing either at the:

- End of all testing – i.e. product Go Live
- End of phase of testing (e.g. hand over from System Test to UAT)

Exit Criteria typically measures:

- Thoroughness measures, such as coverage of requirements or of code or risk coverage
- Estimates of defect density or reliability measures. (e.g. how many defects open by category)
- Cost.
- Residual Risks, such as defects not fixed or lack of test coverage in certain areas.
- Schedules - such as those based on time to market.

10. What determines the level of risk?

Ans. A factor that could result in future negative consequences; usually expressed as impact and likelihood.

- When testing does find defects, the Quality of the software system increases when those defects are fixed.
- The Quality of systems can be improved through Lessons learned from previous projects Analysis of root causes of defects found in other projects can lead to Process Improvement.
- Process Improvement can prevent those defects reoccurring.
- Which in turn, can improve the Quality of future systems.

- **Types of Risk:**

1. Project Risks

2. Product Risks

- **Types of Risk Examples:**

- Example of Project risk is Senior Team Member leaving the project abruptly.
- Example of product risks would be Flight Reservation system not installing in test environment.
- Mitigation in this case would be conducting a smoke or sanity testing. Accordingly, you will make changes in your scope items to include sanity testing.

11. What is Integration testing?

Ans. Integration testing is a type of software testing where individual units or modules of a system are combined and tested as a group. The main goal is to ensure that the units work together correctly and that they interface correctly with each other.

For Example, Imagine you have different parts of a car (like engine, brakes, and steering). Integration testing checks how well these parts fit together and work as a whole.

Types of Integrating Testing:

1. Component Integration Testing: Testing performed to exposed defects in the interfaces and interaction between integrated components.

2. System Integration Testing: It tests the interactions between different systems and may be done after system testing.

12. What is component testing?

Ans. **Component testing,** also known as **unit testing,** is a type of software testing where individual components or modules of a system are tested independently. The goal is to validate that each component behaves as expected when tested in isolation.

- Unit testing is the first level of testing and is performed prior to Integration Testing.

- It's also known as Module Testing or Program Testing.

- Unit testing is performed by using the White Box Testing method.

- Unit tests are typically written and run by **software developers** to ensure that code meets its design and behaves as intended with debugging tool.

Here's a straightforward explanation:

1. Testing Small Parts: Imagine you're building a car. Component testing is like checking each small part of the car, such as the engine or brakes, to make sure they work correctly on their own.

2. Isolation: During component testing, each part is tested independently of the rest of the car. This helps find and fix issues specific to that part without interference from other components.

3. Verification: Developer use specific inputs and compare the actual outputs with expected results to verify that each component performs its intended function accurately.

4. Early Detection of Issues: By testing components early in the development process, problems can be identified and fixed before integrating them into the larger system.

Overall, component testing ensures that individual parts of a software system function correctly on their own before combining them with other parts in integration testing. This approach helps improve the reliability and quality of the final software product.

13. What is functional system testing?

Ans. Functional system testing is a type of software testing that verifies whether a software system or application meets its functional requirements and specifications.

- It focuses on ensuring that each function or feature of the system works as intended, producing the correct output for a given input, without considering the internal implementation details.

14. What is Non-Functional Testing?

Ans. Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance of the software.

- The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters.

- The parameters of non-functional testing are never tested before the functional testing.

- Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

- For example, non-functional testing would be to test how many people can work simultaneously on any software.

15. What is Exploratory Testing?

Ans. Exploratory testing is a flexible and informal approach to software testing where testers actively explore the application without predefined test scripts or formal test cases.

- Instead of following a strict set of instructions, testers use their understanding, experience, and intuition to identify potential issues and evaluate the software's functionality.

16. What is Adhoc testing? (Error Guessing) (Random Testing)

Ans. Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage.

- Ad hoc testing is done **randomly** and it is **usually an unplanned activity** which does not follow **any documentation and test design techniques to create test cases.**

- Main aim of this testing is to find defects by random checking

- Adhoc testing can be achieved with the Software testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

- **Adhoc testing has –** No Documentation, No Test cases, No Test Design.

▪ **Types of Adhoc Testing:**

1. Buddy Testing – Buddy testing is a type of Adhoc testing where two bodies will be involved one is from the Developer team and one from the tester team. So that after completing one module and after completing Unit testing the tester can test by giving random inputs and the developer can fix the issues too early based on the currently designed test cases.

2. Pair Testing – Pair testing is a type of Adhoc testing where two bodies from the testing team can be involved to test the same module. When one tester can perform the random test, another tester can maintain the record of findings. So, when two testers get paired, they exchange their ideas, opinions, and knowledge so good testing is performed on the module.

3. Monkey Testing / Gorilla Testing – Monkey testing is a type of Adhoc testing in which the system is tested based on random inputs without any test cases the behaviour of the system is tracked and all the functionalities of the system are working or not is monitored. As the randomness approach is followed there is no constraint on inputs so it is called Monkey testing.

17. What is white box testing and list the types of white box techniques (coverage)?

Ans. White box testing is a software testing technique that involves testing the internal structure and workings of a software application . The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

it is used to test the software's internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

It is also known as clear box testing, glass box testing, or structural testing.

Types of Coverage:

The different types of coverage are:

- Statement coverage
- Decision coverage
- Condition coverage

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

Ans. Black box testing is a software testing method where the tester does not have knowledge about the internal workings or code structure of the application.
Instead, they focus only on the input and output behaviours, testing the system's functionality based on its specifications, requirements, and user interfaces.
This approach treats the system as a "black box," without considering its internal mechanisms.

Techniques of Black Box Testing:

There are four specification-based or black-box technique:

- Equivalence partitioning
- Boundary value analysis
- Decision tables
- State transition testing
- Use-case Testing
- Other Black Box Testing
    - Syntax or Pattern Testing

19. What is traceability matrix?

Ans. A Traceability Matrix is a document that maps or rather traces the relationship between two baseline documents.

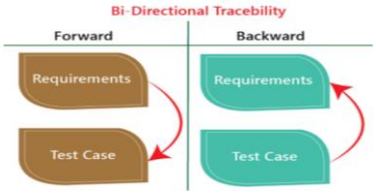
- Here, one of the documents has the requirement specifications, whereas the other one has test cases.
- The purpose of this document is to make sure that all the requirements are covered in test cases so that nothing is missed.
- The Traceability Matrix is an essential document used during the software development lifecycle of a product, and it ensures completeness and transparency of the underlying product.
- It is also called Cross Reference Matrix (CRM) or Requirement Traceability Matrix (RTM).

▪ **Types of Traceability Matrix**

1. Forward Traceability: Mapping of Requirements to Test cases

2. Backward Traceability: Mapping of Test Cases to Requirements

3. Bi-Directional Traceability: A Good Traceability matrix is the References from test cases to basis documentation and vice versa.



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

Ans. Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities. Test plan is a detailed document which describe software testing areas and activity.

▪ **Information to be Covered:**

A Test Plan typically includes the following key aspects:

1. Scope: Outlines the goals of the specific project and provides information about user scenarios intended for testing purposes. It may also specify scenarios or issues that will not be addressed by the project.

2. Test Strategy: Describes the approach to testing, including the types of testing (e.g., functional, regression, usability), testing levels (e.g., unit, integration, system), and testing categories (e.g., black box, white box).

3. Test Deliverables: Specifies the products of testing, such as bug reports, test summaries, and release notes, and outlines the schedule for their creation and submission.

4. Responsibilities: Defines the roles and responsibilities of team members, including test leads, developers, and product managers, and outlines their duties throughout the testing process.

5. Risk and Contingency Planning: Identifies potential risks and threats to the testing process and outlines mitigation strategies and contingency plans.

6. Tools and Environment: Specifies the testing tools, software, and hardware required for testing, as well as the testing environment and infrastructure.

7. Schedule: Outlines the testing timeline, including milestones, deadlines, and critical path activities.

8. Test Coverage: Defines the metrics for measuring test coverage, such as percentage of requirements mapped to test cases.

9. Defect Management: Outlines the procedure for reporting and tracking defects, including the designated recipients and required accompanying elements for each bug report.

21. What is GUI Testing or UI?

Ans. GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.

22. What is load testing?

Ans. In software testing, load testing is an integral part of performance testing under non-functional testing.
Load testing is testing where we check an application's performance by applying some load, which is either less than or equal to the desired load.

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

23. What is stress Testing?

Ans. 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.

Stress Testing is also known as **Endurance Testing.**

A most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks**. It also checks whether the system demonstrates effective error management under extreme conditions.

For Ex. The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives 'Not Responded' error message.

24. Explain types of Performance testing.

Ans. Performance Testing is a type of software testing that ensures software applications 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 is a type of software testing that focuses on evaluating the performance and scalability of a system or application. Performance testing aims to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

The focus of Performance Testing is checking a software program's

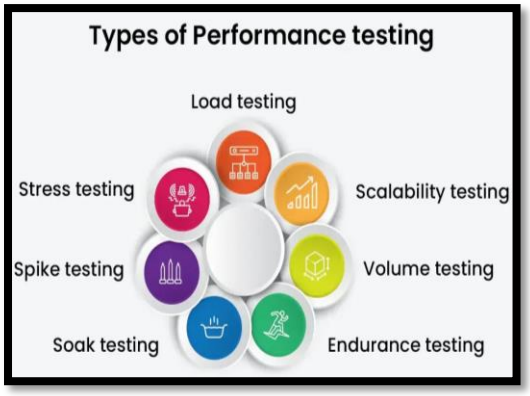
- **Speed** – Determines whether the application responds quickly
- **Scalability** – Determines the maximum user load the software application can handle.
- **Stability** – Determines if the application is stable under varying loads

Why do Performance Testing?

Features and Functionality supported by a software system are not the only concern. A software application's performance, like its response time, do matter.

The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks.

Types of Performance Testing:



1. Load Testing: Its a performance testing to check system behaviour 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.

2. Scalability Testing: The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

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

4. Endurance Testing: it focuses on the long-term behaviour of the system under a constant load. It is performed to ensure the software can handle the expected load over a long period.

5. Soak Testing: A type of load testing that applies a sustained load over an extended period to identify issues that may occur after prolonged usage.

6. Spike Testing: It is a type of load testing that tests the system's ability to handle sudden spikes in traffic. It helps identify any issues that may occur when the system is suddenly hit with a high number of requests. It tests the product's reaction to sudden large spikes in the load generated by users.

7. Stress Testing: involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.

25. When should "Regression Testing" be performed?

Ans. Here are the scenarios when you can apply the regression testing process.

New functionality is added to the application: This happens when new features or modules are created in an app or a website. The regression is performed to see if the existing features are working as usual with the introduction of the new feature.

In case of change requirement: When any significant change occurs in the system, regression testing is used. This test is done to check if these shifts have affected features that were there.

After a defect is fixed: The developers perform regression testing after fixing a bug in any functionality. This is done to determine if the changes made while fixing the bug have affected other related existing features.

Once the performance issue is fixed: After fixing any performance issues, the regression testing process is triggered to see if it has affected other existing functional tests.

While integrating with a new external system: End-to-end regression testing process is required whenever the product integrates with a new external system.

26. What is Alpha testing?

Ans. Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public.

- It is a type of **acceptance testing.**
- The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.
- This testing is referred to as an alpha testing **only because it is done early on, near the end of the development of the software.**
- **Alpha testing is typically performed by in-house software engineers or QA staff. It is the final testing stage before the software is released into the real world.**
- **It comes under the category of both White Box Testing and Black Box Testing.**

27. What is Beta testing/ pre-release testing/ pilot testing/ field testing?

Ans.
- It is always performed by the customers at their own site in the absence of development team.
- It is not performed by Independent Testing Team.
- Beta Testing is always open to the market and public.
- It is usually conducted for software product.
- It is performed in Real Time Environment.
- It is always performed outside the organization.
- It is also the form of **Acceptance Testing.**
- Beta Testing (**field testing**) is performed and carried out by users or you can say people at their own locations and site using customer data.
- It is only a kind of Black Box Testing.
- Beta Testing is always performed at the time when software product and project are marketed.
- It is also considered as the User Acceptance Testing (UAT) which is done at customers or users area.
- Beta testing can be considered "**pre-release**" testing.
- **Pilot Testing** is testing to product on real world as well as collect data on the use of product in the classroom.

28. When to used Usability Testing?

Ans. Usability testing is recommended during the following stages:

1. **Initial design phase**: Conduct usability testing early on to gain visibility into user expectations and identify potential issues with aesthetics and design. This ensures that the product or website is user-cantered from the outset.

2. Before a redesign: Test the existing design to identify areas for improvement and validate assumptions about user needs. This helps prioritize changes and ensures that the redesign meets user requirements.

3. During the redesign: Conduct usability testing throughout the redesign process to validate design decisions, identify potential issues, and make data-driven changes.

4. After a redesign: Test the newly redesigned product or website to ensure that changes have improved the user experience and identify any remaining issues.

5. Ongoing development: Continuously conduct usability testing during ongoing development and optimization to refine the product or website and ensure it remains user-friendly.

29. What is the procedure for GUI Testing?

Ans. The following steps outline the procedure for GUI testing:

1. Test Script Creation

Define a test script template, including:

- Test Script ID, Title (part of functionality under test)
- Test Case ID (links to test cases), Test Setup (environment requirements)
- Test Data (values for usability and correctness checks), Procedure (step-by-step instructions)

2. GUI Element Identification

Identify all graphical user interface elements, including:

- Menus, Checkboxes, Buttons, Colors, Fonts, Sizes, Icons, Content, Images

3. Test Case Development

Create test cases for each GUI element, focusing on:

- Functionality, Usability, Correctness, Error handling

4. Test Data Preparation

Prepare test data values for each test case, including:

- Input values, Expected results, Edge cases

5. Manual Testing

Perform manual testing using the test scripts, focusing on:

- GUI element behavior, User interaction, Error handling, Usability

6. Test Reporting

Document test results, including:

- Pass/Fail status, Defects found, Test environment details

7. Defect Fixing and Re-testing

- Fix defects found during testing, Re-run test cases to ensure defects are resolved

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

Ans. The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating.

Here are the different methodologies in Agile Development Model:

1. Scrum

- **Overview**: SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment.
- Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members).
- It consists of three roles and their responsibilities are explained as follows:
  Scrum Master: Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
  Product owner: The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
  Scrum Team: Team manages its own work and organizes the work to complete the sprint or cycle
- **Key Artifacts**: Product Backlog, Sprint Backlog, Increment
- **Key Events**: Sprint Planning, Daily Scrum (Stand-up), Sprint Review, Sprint Retrospective

2. Kanban

- **Overview**: Kanban is a visual management method that uses boards and cards to represent work items and their status. It focuses on continuous delivery and optimizing workflow.
- **Key Practices**: Visualize work, limit work in progress (WIP), manage flow, make process policies explicit, improve collaboratively.

31. Difference between Smoke and Sanity?

Ans.

No.	Smoke Testing	Sanity Testing
1.	Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality/bugs have been fixed
2.	The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
3.	This testing is performed by the developers or testers	Sanity testing in software testing is usually performed by testers
4.	Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
5.	Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
6.	Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
7.	Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

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

Ans.

Parameter	SDLC	STLC
Origin	Development Life Cycle	Testing Life Cycle
Objective	The main object of SDLC life cycle is to complete successful development of the software including testing and other phases.	The only objective of the STLC phase is testing.
Requirement Gathering	In SDLC the business analyst gathers the requirements and create Development Plan	In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan
High & Low-Level Design	In SDLC, the development team creates the high and low-level design plans	In STLC, the test analyst creates the Integration Test Plan
Coding	The real code is developed, and actual work takes place as per the design documents.	The testing team prepares the test environment and executes them
Maintenance	SDLC phase also includes post-deployment supports and updates.	Testers, execute regression suits, usually automation scripts to check maintenance code deployed.

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

Ans.

Test scenario	Test Cases	Test Script
A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility. Test Scenario is 'What to be tested'	Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. Test Case is 'How to be tested	A set of sequential instruction that detail how to execute a core business function
Test Scenarios are derived from test artifacts like BRS and SRS.	Test Case is mostly derived from test scenarios	Test scripts can also be derived from test scenarios
It is high-level, descriptive, focused on end-to-end functionality, used to guide test case creation	It is manual, human-readable, focused on specific feature, written for manual testing	It is an automated, programming language-based, focused on functionality, written for automated testing
Test Scenarios are high-level descriptions of end-to-end functionality used to guide test case creation.	Test Cases are documents that outline specific testing steps for manual testing.	Test Scripts are sets of automated instructions written in a programming language for automated testing.
Focus : End-to-end functionality	Focus : Specific feature	Focus : Functionality

34. Difference between Sanity and Regression testing?

Ans.

No.	Sanity Testing	Regression Testing
1	Sanity Testing is performed to check the stability of new functionality or code changes in the existing build.	Regression testing is performed to check the stability of all areas impacted by any functionality change or code change.
2	Sanity Testing is part of Regression Testing.	Regression Testing is independent testing.
3	It is executed before Regression Testing and after Smoke Testing.	It is executed based on the project and availability of resources, manpower and time.
4	It examines few functionality of the software.	It examines extended mostly all functionality of the software.
5	Sanity Testing does not use script.	Regression Testing uses Scripts.
6	Sanity Testing is often carried out manually.	Regression Testing is often preferred to continue with automation.
7	This test is shallow and broad.	This test is extensive and in-depth.

35. Difference between retesting and regression testing?

Ans. Retesting means testing the functionality or bug again to ensure the code is fixed. If it is not fixed, defect needs to be re-opened. If fixed, defect is closed.

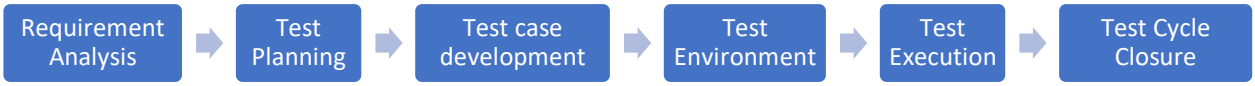
Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

36. STLC life cycle.

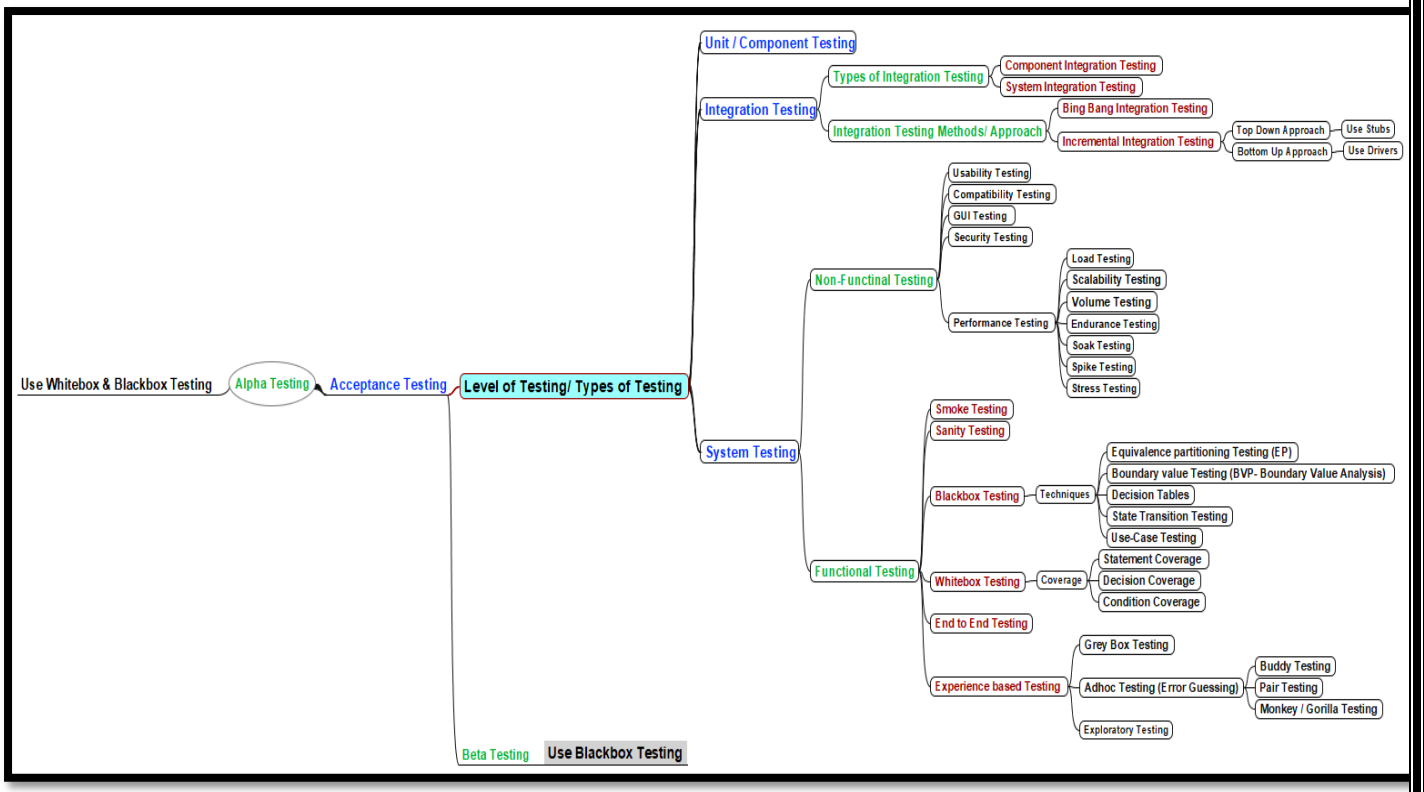
Ans. The Software Testing Life Cycle (STLC) is a systematic approach to testing a software application to ensure that it meets the requirements and is free of defects.

It is a process that follows a series of steps or phases, and each with specific activities and objectives.

The STLC is used to ensure that the software is of high quality, reliable, and meets the needs of the end-users.



PHASES	DISCRIPTION
Requirement Analysis	In this phase, testers understand the testing requirements based on the specifications and requirements documents. They identify what needs to be tested and how.
Test Planning	It is a document describing the scope, approach, resources, and schedule of intended test activities.
Test Case Development	In this phase, involves the creation, verification and rework of test cases & test scripts after the test plan is ready.
Test Environment setup	In this phase, decides the software and hardware conditions under which a work product is tested.
Test Execution	In this phase, the test cases are executed, and the software is tested according to the designed scenarios. Testers record the results and identify any defects or issues encountered.
Test Cycle Closure	The main objective of the test closure phase is to ensure that all testing-related activities have been completed and that the software is ready for release.


Use Whitebox & Blackbox Testing
Alpha Testing
Acceptance Testing
Level of Testing/ Types of Testing
Unit / Component Testing
Integration Testing
Types of Integration Testing
Component Integration Testing
System Integration Testing
Integration Testing Methods/ Approach
Bing Bang Integration Testing
Incremental Integration Testing
Top Down Approach
Use Stubs
Bottom Up Approach
Use Drivers
System Testing
Non-Functinal Testing
Usability Testing
Compatibility Testing
GUI Testing
Security Testing
Performance Testing
Load Testing
Scalability Testing
Volume Testing
Endurance Testing
Soak Testing
Spike Testing
Stress Testing
Functional Testing
Smoke Testing
Sanity Testing
Blackbox Testing
Techniques
Equivalence partitioning Testing (EP)
Boundary value Testing (BVP- Boundary Value Analysis)
Decision Tables
State Transition Testing
Use-Case Testing
Whitebox Testing
Coverage
Statement Coverage
Decision Coverage
Condition Coverage
End to End Testing
Experience based Testing
Grey Box Testing
Adhoc Testing (Error Guessing)
Buddy Testing
Pair Testing
Monkey / Gorilla Testing
Exploratory Testing
Beta Testing
Use Blackbox Testing