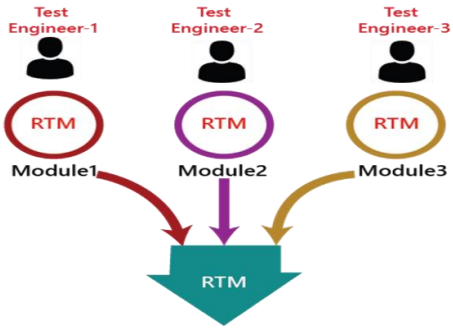
1. What is traceability matrix?

- Traceability matrix is a table type document that is used in the development of software application to trace requirements.
- It can be used for both forward (from Requirements to Design or Coding) and backward (from Coding to Requirements) tracing.
- It is also known as Requirement Traceability Matrix (RTM) or Cross Reference Matrix (CRM)



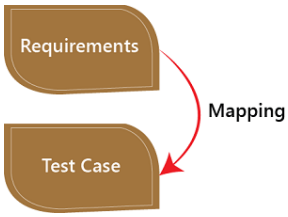
- It is prepared before the test execution process to make sure that every requirement is covered in the form of a Test case so that we don't miss out any testing.
- In the RTM document, we map all the requirements and corresponding test cases to ensure that we have written all the test cases for each condition.

Types of Traceability Test Matrix

- Forward traceability
- Backward or reverse traceability
- Bi-directional traceability

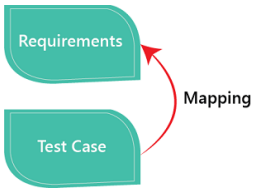
Forward traceability

- The forward traceability test matrix is used to ensure that every business's needs or requirements are executed correctly in the application and also tested rigorously.
- The main objective of this is to verify whether the product developments are going in the right direction.
- In this, the requirements are mapped into the forward direction to the test cases.



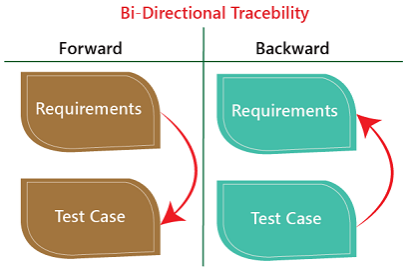
Backward or reverse traceability

- The reverse or backward traceability is used to check that we are not increasing the space of the product by enhancing the design elements, code, test other things which are not mentioned in the business needs.
- And the main objective of this that the existing project remains in the correct direction.
- In this, the requirements are mapped into the backward direction to the test cases.



Bi-directional traceability

- It is a combination of forwarding and backward traceability matrix, which is used to make sure that all the business needs are executed in the test cases.
- It also evaluates the modification in the requirement which is occurring due to the bugs in the application.



Advantage of RTM

- With the help of the RTM document, we can display the complete test execution and bugs status based on requirements.
- It is used to show the missing requirements or conflicts in documents.
- In this, we can ensure the complete test coverage, which means all the modules are tested.
- It will also consider the efforts of the testing teamwork towards reworking or reconsidering on the test cases.

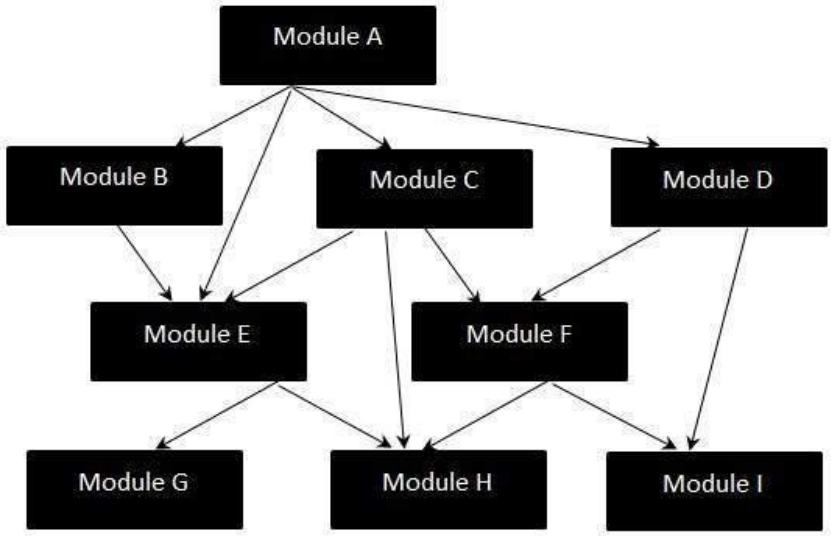
2. What is function system testing?

- **FUNCTIONAL TESTING** is a type of software testing that validates the software system against the functional requirements/specifications.
- The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.
- Functional testing mainly involves black box testing and it is not concerned about the source code of the application.
- This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

3. Mention what bigbang testing is?

- 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 Integration - WorkFlow Diagram



• Disadvantages of Big-Bang Testing

- Defects present at the interfaces of components are identified at very late stage as all components are integrated in one shot.
- It is very difficult to isolate the defects found.
- There is high probability of missing some critical defects, which might pop up in the production environment.
- It is very difficult to cover all the cases for integration testing without missing even a single scenario.

4. What is the purpose of exit criteria?

- Software testing teams will use exit criteria to determine if a test plan or project can exit to the next stage or be considered complete.
- This isn't something that should be left up to the subjective and/or ad hoc decisions of a test admin or SQA engineer, as it can directly impact the success of the next stage or project as a whole.
- Creating exit criteria helps:
- Align your teams on a common definition of test completion
- Ensure your product meets completion standards before entering the next stage, which avoids costly project delays
- Create clear parameters for test engineers to evaluate software.
- Two types of exit criteria
    1. Tester exit criteria
    2. Product exit criteria

5. What is 7 key principles? Explain in details?

➢ **There are seven principles in software testing:**

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

- **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 the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it cannot prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.

- **Exhaustive testing is not possible:** It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test t every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.

- **Early Testing:** To find the defect in the software, early test activity shall be started. The defect detected in the early phases 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.

- **Defect 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.

- **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.

- **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.

- **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 fulfill all the customer requirements.

6. Difference between QA V/S QC V/S Tester.

	QA	QC	Testing
Purpose	Setting up adequate processes ,introducing the standards of quality to prevent the errors and flaws in the product	Making sure that the product corresponds to the requirements and specs before it is released	Detecting and solving software errors and flaws
Focus	Processes	Product as a whole	Source code and design
What	prevention	verification	detection
Who	The team including the stakeholders	The team	Test engineers, developers
When	Throughout the process	Before the release	At the testing stage or along with the development process

7. Difference between smoke and sanity?

Features	Smoke Testing	Sanity Testing
System Builds	Tests are executed on initial builds of software product	Tests are done over builds that have passed smoke tests & rounds of regression tests
Motive of Testing	To measure the stability of the newly created build to face off more rigorous testing	To evaluate rationality & originality of the functionalities of software builds
Subset of?	Is a subset of regression testing	Is a subset of acceptance testing
Documentation	Involves documentation and scripting work	Doesn't emphasize any sort of documentation
Test Coverage	Shallow & wide approach to include all the major functionalities without going too deep	Narrow & deep approach involving detailed testing of functionalities and features
Performed By?	Executed by developers or testers	Executed by testers

8. Difference between verification and validation.

Diff between V&V

Verification	Validation
Are you building it right?	Are you building the right thing?
Ensure that the software system meets all the functionality.	Ensure that functionalities meet the intended behavior.
Verification takes place first and includes the checking for documentation, code etc.	Validation occurs after verification and mainly involves the checking of the overall product.
Done by developers.	Done by Testers.
Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.	Have dynamic activities as it includes executing the software against the requirements.
It is an objective process and no subjective decision should be needed to verify the Software.	It is a subjective process and involves subjective decisions on how well the Software works.

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

➢ Error

- An error is a mistake made by human that leads to discrepancy between the actual and the expected result.

➢ Defect

- A defect is a problem in the functioning of a software system during testing. ISTQB defines a defect as “A flaw in a component or system that can cause the component or system to fail to perform its required function.
- e.g., an incorrect statement or data definition.”

➢ Bug

- A bug is a flaw in a software system that causes the system to behave in an unintended manner.

➢ Failure

- A failure is the inability of a software system to perform its operations within the specified performance benchmark.
- As per ISTQB, “a defect, if encountered during execution, may cause a failure of the component or system”.

10. What is exploratory testing?

- **Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution.

- The focus of exploratory testing is more on testing as a “thinking” activity.
- Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

11. What is boundary value testing ?

- Software testing involves the execution of different testing types to create qualitative and popular among the end-users product.

- But only the main types of testing are well known to all, for example, functional testing, performance checking, black box testing, web application testing, game testing, mobile testing, desktop testing, and so on.

- Equivalence class testing and connected with its boundary value testing are not so well-known. But the funny thing is that these key types of checking are rather often fulfilled by the experts.

- Such a test is unconsciously executed by the testers without the special emphasis on the procedure itself.

- Testing in the equivalence classes always causes a lot of problems. The experienced testers know it very well. So why should one pay special attention exactly to the boundary values?

- This is all about that the boundary values may refer to two classes and, accordingly, the expected result may be different. One of two such results will be a bug. But which one? Firstly, one should puzzle out the reasons for the bugs occurrence during the boundary values input.

12. What is equivalence partitioning testing?

- Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as **ECP.**
- It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.
- An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

13. What is integration testing?

- **Integration Testing** is defined as a type of testing where software modules are integrated logically and tested as a group.
- A typical software project consists of multiple software modules, coded by different programmers.
- The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated
- Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes **'Thread Testing'**.

Types of Integration Testing

- Big Bang Approach :
- Incremental Approach: which is further divided into the following
    - Top Down Approach
    - Bottom Up Approach
    - Sandwich Approach – Combination of Top Down and Bo

14. What is component testing?

- **Unit Testing** is a type of software testing where individual units or components of a software are tested.
- The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness.
- A unit may be an individual function, method, procedure, module, or object.
- In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer.
- Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

15. What is non-function testing?

- Non-functional testing is testing designed to check a software application's non-functional aspects such as performance, reliability, utility,

- etc. In addition, this testing is designed to test a system's readiness related to non-functional parameters typically never addressed by the functional testing process.

- For instance, how many users can simultaneously log in to a software application? Or how easy or difficult is it to port the application to a different system? Or does the application behave differently in another operating system or environment?

- So non-functional testing puts an application through its paces, testing aspects that functional testing doesn’t touch.

16. What is GUI testing?

- **UI 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.
- GUI is what the user sees. Say if you visit guru99.com what you will see say homepage it is the GUI (graphical user interface) of the site.
- A user does not see the source code. The interface is visible to the user.
- Especially the focus is on the design structure, images that they are working properly or not.
- In above example, if we have to do GUI testing we first check that the images should be completely visible in different browsers.
- Also, the links are available, and the button should work when clicked.
- Also, if the user resizes the screen, neither images nor content should shrink or crop or overlap.

17. What is Adhoc testing?

- **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.

- Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
- 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.
- This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

18. **What is load testing**?

- **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.

- his testing usually identifies –

- The maximum operating capacity of an application.
- Determine whether the 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.

- It is a type of non-functional testing. In Software Engineering, Load testing is commonly used for the Client/Server, Web-based applications – both Intranet and Internet.

19. What is stress Testing?

- **Stress Testing** is a type of software testing that verifies stability & reliability of software application.
- The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations.
- It even tests beyond normal operating points and evaluates how software works under extreme conditions.

- In Software Engineering, Stress Testing is also known as Endurance Testing.
- UnderStress Testing, AUT is be stressed for a short period of time to know its withstanding capacity.
- 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.
- 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.

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

- **White Box Testing** is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security.
- In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.
- It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective.
- On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.
- The term "WhiteBox" was used because of the see-through box concept.
- The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings.
- Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

List the type white box testing

- **Test coverage**

  - Statement Coverage
  - Decision Coverage
  - Condition Coverage

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

- **Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.
- Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications.
- It is also known as Behavioral Testing.

- The above Black-Box can be any software system you want to test.
- For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application.
- Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation. Consider the following video tutorial

➢ Different black box testing techniques

1. Equivalence Partitioning

- Testers can divide possible inputs into groups or “partitions”, and test only one example input from each group.

- For example, if a system requires a user’s birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the “under 18” group and one date in the “over 18” group.

2. Boundary Value Analysis

- Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99.

- Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

3. Decision Table Testing

- Many systems provide outputs based on a set of conditions.

- Testers can then identify “rules” which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

- For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not.

- This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.

4. State Transition Testing

- In some systems, significant responses are generated when the system transitions from one state to another.

- A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.

- If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states.

- For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.

5. Graph-Based Testing:

- It is similar to a decision-based test case design approach where the relationship between links and input cases are considered.

6. Error Guessing Technique:

- This method of designing test cases is about guessing the output and input to fix any errors that might be present in the system. It depends on the skills and judgment of the tester.

22. When should "Regression Testing" be performed?

- Ideally, regression testing should be performed whenever your codebase has been modified or altered in any way as well as to verify any previously discovered issues marked as fixed.
- The more often the better: frequent partial regression testing will help your developers fix the reported defects on time, and your project to avoid any long-term pitfalls and technical debt caused by poor code quality.
- However, even though an occasional project might have the resources to perform the tests after the slightest changes have been introduced to the codebase, for most projects designing and maintaining such a multiplicity of regression tests may simply be infeasible.
- Therefore, it is important to understand when you need to start regression testing.
- The most common reason to run regression tests is the introduction of **new functionality**. It is hard for developers to follow every thread in the code when modifying it, and there's always a risk of compatibility issues with the existing code.
- Regression testing can save developers a lot of time with timely detection of bugs that would otherwise cause the project a lot of pain in the long run.
- Sometimes, however, sudden shifts in business strategy and requirements can lead to complete **revision of the existing functionality**, which requires developers to adjust, reshape, or even discard some of the features.
- Heavy interference with the source code can potentially cause a lot of damage to the remaining functionality, which makes regression testing an absolute must in such cases.
- Attempts at fixing one bug can at times turn into even more bugs appearing in codebase areas you expect them the least.
- **Debugging** suggests making a lot of big and small changes to the source code, checking bug statuses, and going through the same process over again.

- This, in turn, points out the importance of following up the debugging stage with regression testing to guarantee it didn't cause more issues than it fixed; to make sure everything works as intended after debugging.
- Last but not least, regression testing has proven to be an effective means of ensuring seamless and bug-free **integration with external systems**.

23. Explain types of Performance testing.

➢ Types of Performance Testing

- **Volume testing** - The main objective of volume testing is to check the performance of the application in different database volumes. The behavior of the application is monitored by populating varying volumes of data into the database.
- **Stress testing** - The main objective of stress testing is to identify the main breaking point of a software application. This is done by testing the application under extreme workloads to gauge its performance under high traffic or data processing.
- **Spike testing** - The main objective of spike testing is to test the reaction of the application when a sudden large spike (generated by users) occurs in the load.
- **Scalability testing** - The main objective of scalability testing is to determine whether the application can scale up effectively in the event of user overload. This testing also helps you to plan capacity addition to your application for the future.
- **Load testing** - The main objective of load testing is to identify performance bottlenecks or the application's ability to perform under anticipated user loads.
- **Endurance testing** - Endurance testing is done to make sure the software can handle the expected load over a long period of time.

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

Difference Between the Functional and non - functional testing

Functional Testing	Non-Functional Testing
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 non-functional aspects of the software system.
Functional testing is executed first	Non functional testing should be performed after functional testing
Manual testing or automation tools can be used for functional testing	Using tools will be effective for this testing
Business requirements are the inputs to functional testing	Performance parameters like speed , scalability are inputs to non-functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Easy to do manual testing	Tough to do manual testing
Types of Functional testing are ✔ Unit Testing ✔ Smoke Testing ✔ Sanity Testing ✔ Integration Testing ✔ White box testing ✔ Black Box testing ✔ User Acceptance testing ✔ Regression Testing	Types of Non functional testing are ✔ Performance Testing ✔ Load Testing ✔ Volume Testing ✔ Stress Testing ✔ Security Testing ✔ Installation Testing ✔ Penetration Testing ✔ Compatibility Testing

25. What determines the level of risk?

- **Low Risk**
  - System processes and/or stores public data
  - System is easily recoverable and reproducible
  - System provides an informational / non-critical service
- **Moderate Risk**
  - System processes and/or stores non-public or internal-use data
  - System is internally trusted by other networked systems
  - System provides a normal or important service
- **High Risk**
  - System processes and/or stores confidential or restricted data
  - System is highly trusted by UI networked systems
  - System provides a critical or campus-wide service

- Risk Analysis must take into consideration the sensitivity of data processed and stored by the system, as well as the likelihood and impact of potential threat events.
- We use a simple methodology to translate these probabilities into risk levels and an overall system risk level.

26. What is Alpha testing?

- **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, and before Beta Testing. Check Differences between Alpha testing and Beta testing
- 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.
- Central environment create

27. What is beta testing?

- **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. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation.

- It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

- Beta testing is a type of user acceptance testing where the product team gives a nearly finished product to a group of target users to evaluate product performance in the real world.

- Real environment create

28. Mention what are the categories of defects?

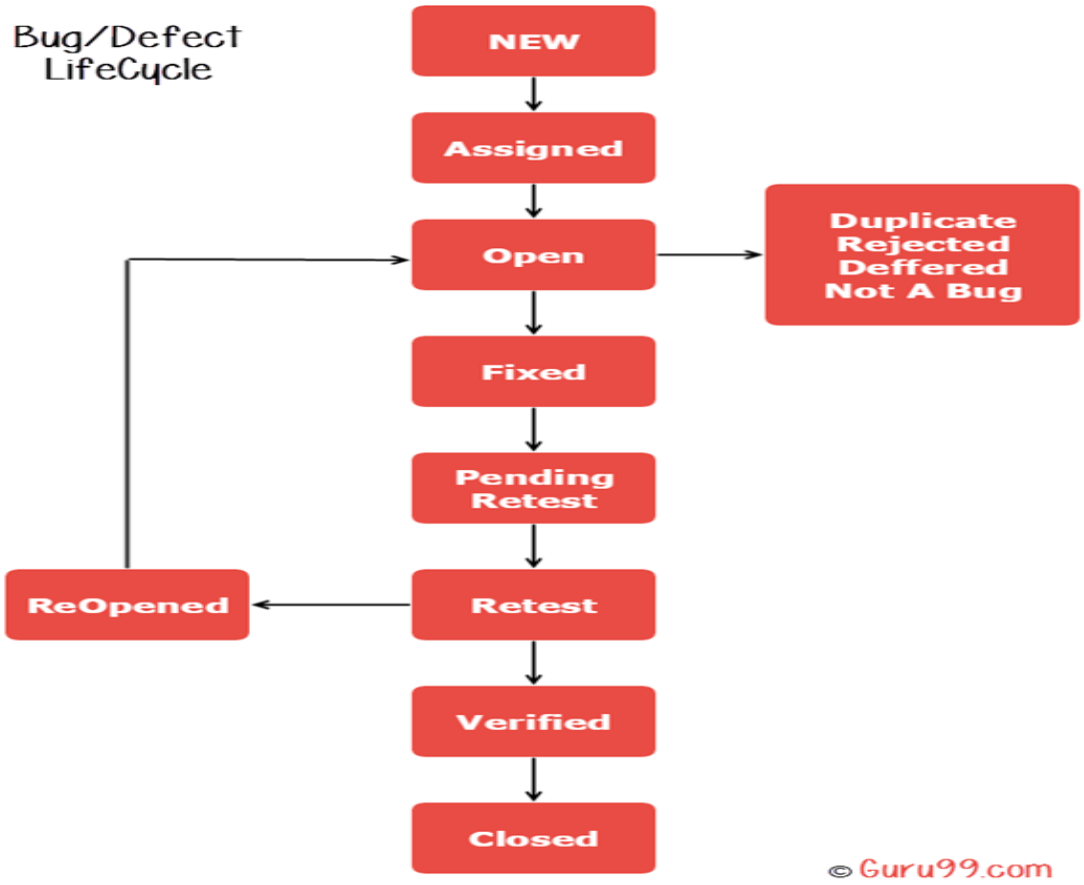
- **Wrong:** This defect category is a variance from the given requirement specification. This means if there are any discrepancies in requirement documents it will fall under this category.

- **Missing:** During the development or design if we miss any requirement which is given by the customer those type of defect categories fall under this. This type of missing causes by when requirement explain by the customer was not noted properly and customer requirement not explained to the team properly when they were developed.

- **Extra:** During the development if a new requirement added to the existing requirements which is not given by the customer those changes or defects fall under this category.

29. What is bug life cycle?

- **The bug life cycle** is also known as the **Defect life cycle**. In the Software Development Process, the bug has a life cycle.
- The bug should go through the life cycle to be closed. The bug life cycle varies depends upon the tools (QC, JIRA, etc.,) used, and the process followed in the organization.

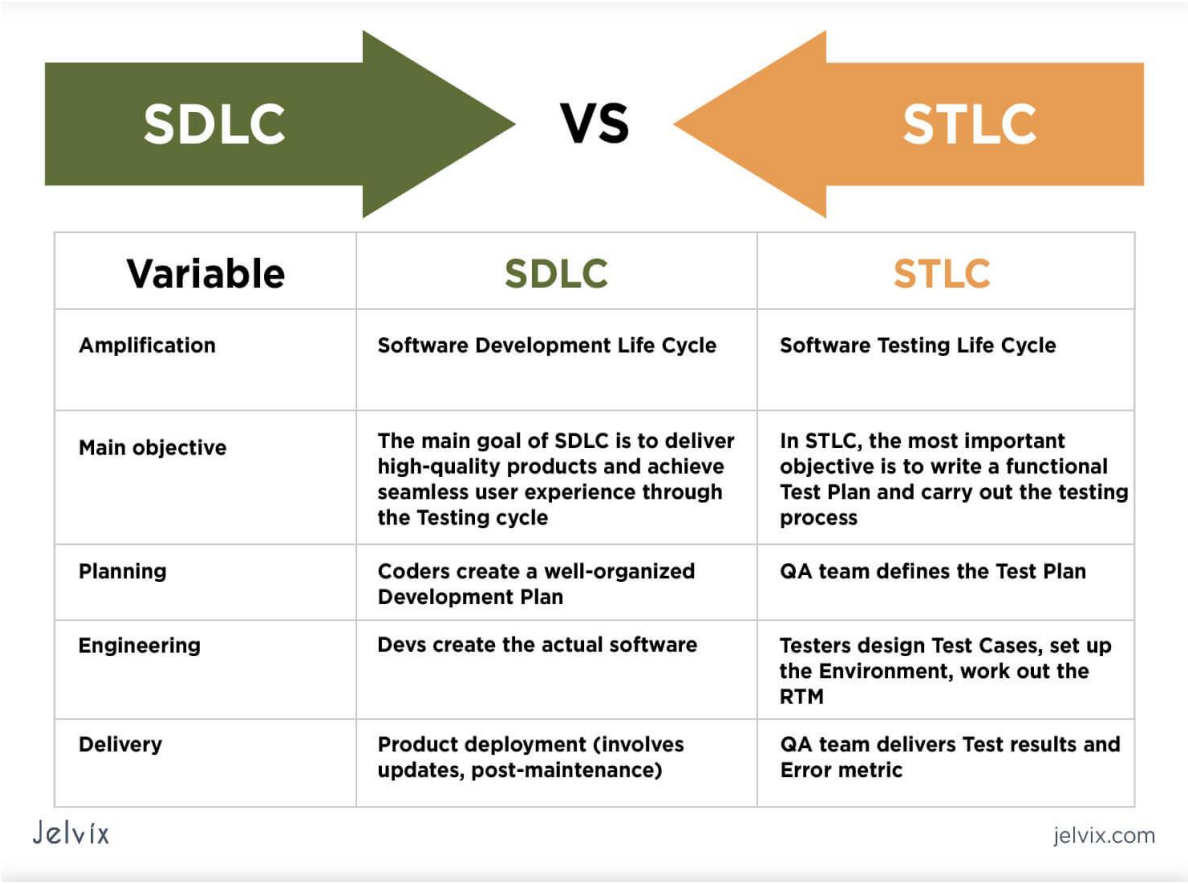


- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
- **Open**: The developer starts analyzing and works on the defect fix
- **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."
- **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is "pending retest."
- **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test."

- **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."

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

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



31. What is the difference between test Scenarios, test cases, and test script?

Test Scripts

- This story begins with the most detailed way to document testing, the test script. When people talk about test scripts, they usually mean a line-by-line description of all the actions and data needed to perform a test. A script typically has ‘steps’ that try to fully describe how to use the program — which buttons to press, and in which order — to carry out a particular action in the program. These scripts also include specific results that are expected for each step, such as observing a change in the UI. An example step might be “Click the ‘X’ button,” with an example result of “The window closes.”
- When a tester first starts a new job, they might not know much about the product, the business domain, or even software testing. Scripts can help bridge that gap. If the tester carefully follows the directions — enter the string ‘abc’, click the submit button, make sure the form submitted and the value was saved — the test idea will be covered enough to consider it ‘tested’.
- There are a few drawbacks to consider before going all-in with detailed scripts. Active software projects change often — pages get redesigned, user experience changes, and new functionality is added. To be effective over time, testers have to make a continuous effort to update the scripts to match the new product. This can take time away from testing. Another drawback is that scripted tests are often designed to test one specific thing repeatedly, using the same steps and the same data each time the test is executed. This means that if there are bugs that lie outside the directions given in the test script, they will not be found unless the tester strays from the script. Scripted tests do not always encourage testers to use the creativity and technical skill required to find hidden bugs.

Test Cases

- The second most detailed way of documenting testing work is to use test cases. Test cases describe a specific idea that is to be tested, without detailing the exact steps to be taken or data to be used. For example, a test case might say “Test that discount codes can be applied on top of a sale price.” This doesn’t mention how to apply the code or whether there are multiple ways to apply the code. The actual testing that will cover this test case may vary from time to time. Will the tester use a link to apply a discount, or enter a code, or have a customer service rep apply the discount, or will they feel compelled to test every way to add a discount that they can think of? Test cases give flexibility to the tester to decide exactly how they want to complete the test.
- This flexibility from test cases is both good and bad. Flexibility is beneficial when the tester is familiar with testing and familiar with the software under test and the current set of risks in the software. If the tester clearly understands what has already been tested, what has changed recently in the program, and how users typically use the program, they will choose an approach in their testing that will exercise both the most common user paths, and the less common paths that are most likely to reveal bugs.
- On the other hand, if the tester does not have a good understanding of how the program is used, the recent risks to the program, and how to evaluate those risks as a tester, they may not have the information or skill they need to assess the actions required to reveal important bugs.

Test Scenarios

- The least detailed type of documentation is the test scenario. A test scenario is a description of an objective a user might face when using the program. An example might be “Test that the user can successfully log out by closing the program.” Typically, a test scenario will require testing in a few different ways to ensure the scenario has been satisfactorily covered. Just based on that light description, the tester might choose to close the program through the menu option, kill it through the task manager, turn the computer off, or see what happens when the program runs out of memory and crashes. Since test scenarios offer little information about how to complete the testing, they offer the maximum amount of flexibility to the tester responsible for them.
- Like test cases, the flexibility that comes with using test scenarios creates similar benefits and drawbacks. Testing skill and domain knowledge make it easier for the tester to break test scenarios down into the relevant test ideas, select the approach that makes most sense, and perform tests that find important problems. This work is fun and challenging for a skilled tester, but it may be difficult or impossible for a novice unless they are able to collaborate with others to get the needed skill and perspective.

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

- In general, testing commences with a test plan and terminates with acceptance testing. A test plan is a general document for the entire project that defines the scope, approach to be taken, and the schedule of testing as well as identifies the test items for the entire testing process and the person responsible for the different activities of testing.
- The test planning can be done well before the actual testing commences and can be done in parallel with the coding and design phases. The inputs for forming the test plan are: (1) project plan (2) requirements document and (3) system design document. The project plan is needed to make sure that the test plan is consistent with the overall plan for the project and the testing the test plan is consistent with the overall plan for the project and the testing schedule matches that of the project plan.
- The requirements document and the design document are the basic documents used for selecting the test units and deciding the approaches to be used during testing. A test plan should contain the following:

➢ **Test unit specification**
Features to be tested
Approach for testing
Test deliverable
Schedule
Personnel allocation

- One of the most important activities of the test plan is to identify the test units. A test unit is a set of one or more modules, together with associate data, that are from a single computer program and that are the objects of testing. A test unit can occur at any level and can contain from a single module to the entire system.
- Thus, a test unit may be a module, a few modules, or a complete system. The levels are specified in the test plan by identifying the test units for the project. Different units are usually specified for unit integration, and system testing. The identification of test units may be a module, a few modules or a complete system. The levels are specified in the test plan by identifying the test units for the project.
- Different units are usually specified for unit, integration and system testing. The identification of test units establishes the different levels of testing that will be performed in the project. The basic idea behind forming test units is to make sure that testing is being performed incrementally, with each increment including only a few aspects that need to be tested. A unit should be such that it can be easily tested.

33. Bug categories are…

➢ Software Bugs by Nature:

- Software bugs have different natures where they affect the overall functioning of the software differently. Though there are dozens of such bugs existing currently, you may not face them frequently. With that in mind, here are the most common software bugs categorized by nature that you are most likely to witness at some point in your software development career.

• Performance Bugs:

- No user wants to use software with poor performance. Software bugs that lead to degraded speed, stability, increased response time, and higher resource consumption are considered performance bugs. The most significant sign of any such bug in software is by noticing slower loading speed than usual or analyzing the response time. If any such sign is found, the developer may begin diagnosing a performance bug. The performance testing phase is part of the development process where every such bug is detected in the software.

• Security Bugs:

- While using software, security is the biggest concern of a user. Software with poor security will not only put the user's data at risk but will also damage the overall image of the organization which may take years to recuperate. Due to their high severity, security bugs are considered among the most sensitive bugs of all types. Though it is self-explanatory, security bugs may make the software vulnerable to potential cyber threats. Sometimes, the software organization may not notice such attacks whereas in some cases, these attacks could cause monetary loss to the users, especially small and medium-scale businesses. XSS vulnerabilities, logical errors, and encryption errors are some of the commonest security bugs found in the software. Developers put special focus on checking the code to find any underlying security bug to minimize the risk of cyber-attacks.

• Unit Level Bugs:

-

Unit level bugs are fairly common in software development and do not cause much damage to it as well. Facing basic logic bugs or calculation errors are considered unit-level bugs. The testing team along with the agile team test a small part of the code as a whole. The reason why this testing method is preferred is to make sure that the entire code is working as it is meant to. While testing, the team may encounter unit-level bugs which can be fixed easily as the team is only working with a small code.

○ Functional Bugs:

- Software is as good as the feature it provides. If any of the functionality of a software is compromised, the number of users will start to decline drastically until it becomes functional again. A functional bug is when a certain feature or the entire software is not functioning properly due to an error. The severity of such bugs depends on the feature they are hampering. For instance, an unresponsive clickable button that is not functioning is not as severe as the entire software not working. Functional testing is done by the testing team to identify any such software bug causing functionality errors. Once identified, the team decides its further classification and severity.

○ Usability Bugs:

- Probably one of the most catastrophic bugs for software, a usability bug or defect can stop the software from working to its potential or make it entirely unusable. Examples of this bug in software testing are the inability to log in to the user account or the inefficient layout of the software for the user. The bottom line is that this type of defect or bug can make it complex for the user to use the software efficiently. The developers and engineers have to look out for the right usability requirements while testing the code to identify such bugs.

○ Syntax Errors:

- Syntax errors are among the commonest software bug types and do not allow the application to be compiled appropriately. This bug occurs due to an incorrect or missing character from the source code due to which the compiling will be affected. A small error like a missing bracket could lead to this problem. The development or testing team will get to know about this bug during compiling and will further analyse the source code to fix the missing or wrong characters.

○ Compatibility Errors:

- Whenever a software or an application is not compatible with hardware, or an operating system, it is considered as incompatible software or a compatibility error. Finding a compatibility error is not a common practice as they may not show up in the initial testing. Due to this reason, the developers should go for compatibility testing to make sure that their created software is compatible with common hardware and operating systems.

○ Logic Bugs:

- Another one of the most frequently found bugs in a software code, logic errors make the software give wrong output, software crash or failure. In the majority of cases, these bugs are caused due to coding errors where it may make the software stuck in a never-ending loading loop. In that case, only an external interruption or software crashing are the two only things that can break the loading loop.

➢ Priority-Based Software Bugs:

- The foremost category here is priority-based software bugs. These are based on the impact these bugs leave on the business. Here, the developers will analyze the bug to determine its impact and its defect priority. Afterward, the timeline is given to each bug where it should be rectified within the stipulated time frame to minimize the bug effect on the user. Here are the four types of priority-based software bugs.

• Low-priority defects:

- Low priority defects do not cause much impact on the functioning of the application. Rather, they are more about software aesthetics. For instance, any issue with the spelling or the alignment of a button or text could be a low-priority defect. The software testing will move to the exit criteria even if the low-priority defects are not fixed, but they should be rectified before the final release of the software.

• Medium-priority defects:

- Akin to low-priority defects, medium-priority defects do not cause any significant impact on the software, but they should be fixed in any subsequent or upcoming release. Such defects may not have the same effect for every user and it may vary with the device as well as specific configuration they have

• High-priority defects:

- Unlike the previous two, the exit criteria of high-priority defects are not met until the issue is resolved. Every bug falling in this category may make certain features of the software unusable. Even though it may not affect every user, it is mandatory to fix these bugs before any further step is taken in software development or testing

• Urgent Defects:

- As the name suggests, all bugs that should be dealt with utmost urgency fall under this category. Urgent defects may leave a lasting impact on the brand image as well as affect the user experience drastically. The stipulated timeline for fixing these bugs is within 24-hours of reporting.

34. When to used Usability Testing?

1. Guerilla Testing

- It is a type of testing where testers wander to public places and ask random users about the prototype.
- Also, a thank gift is offered to the users as a gesture of token. It is the best way to perform usability testing during the early phases of the product development process. Users generally spare 5-10 minutes and give instant feedback on the product.
- Also, the cost is comparatively low as you don't need to hire participants. It is also known as corridor or hallway testing.

2. Usability Lab

- Usability lab testing is conducted in a lab environment where moderators (who ask for feedback on the product) hire participants and ask them to take a survey on the product. This test is performed on a tablet/desktop.
- The participant count can be 8-10 which is a bit costlier than Guerilla testing as you need to hire participants, arrange a place, and conduct testing.

3. Screen or Video Recording

- Screen or video recording kind of testing is in which a screen is recorded as per the user's action (navigation and usage of the product).
- This testing describes how the user's mind runs while using a product.
- This kind of testing involves the participation of almost 10 users for 15 minutes. It helps in describing the issues users may face while interacting with the product.

35. What is the procedure for GUI Testing?

1. **Manual Testing**
   - Testing is manually conducted to verify that graphical screens align with the stipulated business specifications.
2. **Automation**
   - Testing Testing is automated with programs that help to record and replay tests. We first must register initial test procedures and subsequently replay them in the software being tested.
3. **Model-based Testing**