Module-2 (Manual Testing)

1. What is Exploratory Testing?

- If requirement does not exist, then we do one round of exploratory testing.
- So, for this first, we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application, this approach is known as exploratory testing.

➔ **When we use Exploratory testing**

- When the requirement is missing
- Early iteration is required
- The testing team has the experienced testers when we have a critical application, and new testers entered into the team.

➔ **Example**

- To test any software or the application, first, we will perform unit, integration, and system testing.
- So if we want to understand any application first, we will perform unit or component testing, suppose the application is having a login page having many elements, and we will understand each part and doing the component testing, but actually, we are doing the exploratory testing because we are exploring the application.
- Suppose we have many modules in the application, and we are trying to do some integration scenarios.
- Indirectly we are just doing exploratory testing while performing the integration testing.
- And, even if we are performing system testing, indirectly, we are performing exploratory testing because here we are also understanding and exploring the application.

2. 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.
- The test engineer will prepare RTM for their respective assign modules, and then it will be sent to the Test Lead. The Test Lead will go repository to check whether the Test Case is there or not and finally Test Lead consolidate and prepare one necessary RTM document.
- This document is designed to make sure that each requirement has a test case, and the test case is written based on business needs, which are given by the client. It will be performed with the help of the test cases if any requirement is missing, which means that the test case is not written for a particular need, and that specific requirement is not tested because it may have some bugs. The traceability is written to make sure that the entire requirement is covered.

3. What is boundary value testing?

- Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.
- Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.
- Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.
- The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.
- There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

4. What is equivalence partitioning testing?

- Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behaviour. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behaviour as other.
- The equivalence partitions are derived from requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.

Example

- Assume that there is a function of a software application that accepts a particular number of digits, not greater and less than that particular number. For example, an OTP number which contains only six digits, less or more than six digits will not be accepted, and the application will redirect the user to the error page.

5. What is integration testing?

- Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
- Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
- The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- Integration testing is done by a specific integration tester or test team.
- Components may be code modules, operating systems, hardware and even complete systems.

- There are two levels of Integration testing
    - Component Integration Testing
    - System Integration Testing

6. What determines the level of risk?

- Once potential risks have been identified in the software testing process, next it is important to determine the level of this risk – this process is called risk analysis and risk-level appointment.
- Determining the level of risk usually involves trying to assess not only the likelihood of an identified risk from actually occurring, but also the potential magnitude the consequences this risk could have on an organisation and its stakeholder, should it occur. Both likelihood of occurrence and potential magnitude of impact can be measured through the frequency of occurrence and simulated impact that can be observed under test conditions. As mentioned above, influencing factors when it comes to product and project risks include the presence of complex technology, legacy issues, general programming problems, personnel and training issues among software developers and testers, and unforeseen delays to the project as a whole.
- Once risks are identified through testing and analysis, they can then be assigned a 'level of risk' based on a set of pre-conceived criteria. For example, 'High', 'Medium' and 'Low' risks.

7. What is alpha testing?

- It is always performed by the developers at the software development site.
- Sometimes it is also performed by Independent Testing Team.
- Alpha Testing is not open to the market and public
- It is conducted for the software application and project.
- It is always performed in Virtual Environment.
- It is always performed within the organization.
- It is the form of Acceptance Testing.
- Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers.
- It comes under the category of both White Box Testing and Black Box Testing.

8. What is beta testing?

- It is always performed by the customers at their own site.
- 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.

9. What is component testing?

- A minimal software item that can be tested in isolation. It means "A unit is the smallest testable part of software."
- Component Testing – The testing of individual software components.
- Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
- Unit testing is the first level of testing and is performed prior to Integration Testing.
- Sometimes known as Unit Testing, Module Testing or Program Testing
- Component can be tested in isolation – stubs/drivers may be employed
- Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing.
- Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.
- A unit is the smallest testable part of an application like functions/procedures, classes, interfaces.
- The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.
- A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.
- Unit tests find problems early in the development cycle.
- Unit testing is performed by using the White Box Testing method.

10. What is functional system testing?

- Functional Testing: Testing based on an analysis of the specification of the functionality of a component or system.
- Specification – E.g. Requirements specification, Use Cases, Functional specification or maybe undocumented.
- Function – what the system does
- Functional test – based on the Functions and features – may be applied at all Test levels (e.g. Component Test, System Test etc.)
- Considers the external (not internal) behaviour of the software. Black-Box testing. What it does rather than how it does it. More on this later!
- Functional testing verifies that each function of the software application operates in conformance with the requirement specification.
- This testing mainly involves black box testing and it is not concerned about the source code of the application.
- Each & every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
- This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application under Test. The testing can be done either manually or using automation

➔ Functional Testing Example

- Web Based Testing
- Desktop Based Testing
- Mobile Based Testing
- Game Based Testing

11. What is non-functional testing?

- Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, interoperability, maintainability and portability
- May be performed at all Test levels (not just Non Functional Systems Testing)
- Measuring the characteristics of the system/software that can be quantified on a varying scale- e.g. performance test scaling
- Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.
- It is the testing of “how” the system works. Non-functional testing may be performed at all test levels.

- The term non-functional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale, such as response times for performance testing.
- To address this issue, performance testing is carried out to check & fine tune system response times. The goal of performance testing is to reduce response time to an acceptable level
- Hence load testing is carried out to check systems performance at different loads i.e. number of users accessing the system

➔ Non Functional Testing Example

- Web Based Testing
- Desktop Based Testing
- Mobile Based Testing
- Game Based Testing

12. What is GUI Testing?

- Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.
- WHAT DO YOU CHECK IN GUI TESTING?
- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the Colour of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

➔ Approach of GUI testing

- Manual Based Testing
- Record and Replay
- Model Based Testing

➔ **Example**

- Web based testing and GUI based testing
- Mobile based testing
- Game based testing

13. What is Adhoc testing?

- Adhoc testing is an informal testing type with an aim to break the system.
- It does not follow any test design techniques to create test cases.
- In fact is does not create test cases altogether!
- This testing is primarily performed if the knowledge of testers in the system under test is very high.
- Testers randomly test the application without any test cases or any business requirement document.
- Adhoc 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 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.
- The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.
- Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.
- This is why an error guessing approach, used after more formal techniques have been applied to some extent, can be very effective.

14. What is load testing?

- It's 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.
- This testing usually identifies –

    - The maximum operating capacity of an application
    - Determine whether current infrastructure is sufficient to run the application
    - Sustainability of application with respect to peak user load
    - Number of concurrent users that an application can support, and scalability to allow more users to access it.
    - It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.
- Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to ensure optimal system performance, when that marketing brings in traffic.

➔ Example

- Popular toy store Toysrus.com, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.
- An Airline website was not able to handle 10000+ users during a festival offer.
- Encyclopaedia Britannica declared free access to their online database as a promotional offer. They were not able to keep up with the onslaught of traffic for weeks.
- Facebook(FB)

➔ Load Testing Tools

- Loadrunner
- Web Load
- Astra Load Test
- Review's Web Load
- Studio, Rational Site Load
- Silk Performer

➔ Pros of Load testing

- Performance bottlenecks identification before production
- Improves the scalability of the system
- Minimize risk related to system down time
- Reduced costs of failure
- Increase customer satisfaction

➔ Cons of Load testing

- Need programming knowledge to use load testing tools.
- Tools can be expensive as pricing depends on the number of virtual users supported.

15. What is stress testing?

- System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
- Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions**.**
- It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
- Stress Testing is done to make sure that the system would not crash under crunch situations.
- Stress testing is also known as endurance testing.
- Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.
- It also checks whether 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.

➔ Example

- Excessive volume in terms of either users or data; examples might include a denial of service (DoS) attack or a situation where a widely viewed news item prompts a large number of users to visit a Web site during a three-minute period.
- Resource reduction such as a disk drive failure.
- Application components fail to respond.

➔ Types of Stress testing

- Application Stress Testing
- Transactional Stress Testing
- Systemic Stress Testing
- Exploratory Stress Testing

➔ **Stress Testing tools**

- Stress tester
- Neo Load
- App Perfect

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

- White Box Testing: Testing based on an analysis of the internal structure of the component or system.
- Structure-based testing technique is also known as 'white-box' or 'glass-box' testing technique because here the testers require knowledge of how the software is implemented, how it works.
- In white-box testing the tester is concentrating on how the software does it.
- For example, a structural technique may be concerned with exercising loops in the software.
- Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
- Structure-based techniques are also used in system and acceptance testing, but the structures are different.
- For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.
- Testing based upon the structure of the code
- Typically undertaken at Component and Component Integration Test phases by development teams
- White box testing is the detailed investigation of internal logic and structure of the code.
- White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

➔ **White Box testing type**

- Statement Coverage
- Decision Coverage
- Condition Coverage

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

- Testing, either functional or non-functional, without reference to the internal structure of the component or system.
- Specification-based testing technique is also known as 'black-box' or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
- The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.
- Specification-based techniques are appropriate at all levels of testing(component testing through to acceptance testing) where a specification exists.
- For example, when performing system or acceptance testing, the requirements specification or functional specification may form the basis of the tests.
- The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.
- What a system does, rather than HOW it does it
- Typically used at System Test phase, although can be useful throughout the test lifecycle
- The tester is oblivious to the system architecture and does not have access to the source code.
- Typically, when performing a black box test**,** a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

➔ <u>Black box testing types</u>

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table
- State Transition
- Use Case

18. Mention what are the categories of defects?

-

a) Data Quality/Database Defects

- Deals with improper handling of data in the database.
- **<u>Examples</u>**: Values not deleted/inserted into the database properly

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

b) Critical Functionality Defects

- The occurrence of these bugs hampers the crucial functionality of the application.
- **Examples**: - Exceptions

c) Functionality Defects

- These defects affect the functionality of the application.
- **Examples:**
- All JavaScript errors
- Buttons like Save, Delete, Cancel not performing their intended functions
- A missing functionality (or) a feature not functioning the way it is intended to Continuous execution of loops

d) Security Defects

- Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.
- **Examples**:
- Authentication: Accepting an invalid username/password
- Authorization: Accessibility to pages though permission not given

e) User Interface Defects

- As the name suggests, the bugs deal with problems related to UI are usually considered less severe.
- **Examples**:
- Improper error/warning/UI messages
- Spelling mistakes
- Alignment problems

19. Mention what bigbang testing is?

- In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
- Big Bang testing has the advantage that everything is finished before integration testing starts.

- The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
- Here all component are integrated together at **once**, and then tested.

➔ Advantages

- Convenient for small systems.

➔ Disadvantages

- Fault Localization is difficult.
- Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
- Since the integration testing can commence only after "all" the modules are designed, testing team will have less time for execution in the testing phase.
- Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

20. What is the purpose of exit criteria?

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

➔ <u>Example</u>

- Verify if All tests planned have been run.
- Verify if the level of requirement coverage has been met.
- Verify if there are NO Critical or high severity defects that are left outstanding.
- Verify if all high risk areas are completely tested.
- Verify if software development activities are completed within the projected cost.
- Verify if software development activities are completed within the projected timelines.

21. When should regression testing be performed?

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

- If the test is re-run and passes you cannot necessarily say the fault has been resolved because You also need to ensure that the modifications have not caused unintended side-effects elsewhere and that the modified system still meets its requirements – Regression Testing

➔ **Regression testing should be carried out**

- when the system is stable and the system or the environment changes
- when testing bug-fix releases as part of the maintenance phase
- It should be applied at all Test Levels
- It should be considered complete when agreed completion criteria for regression testing have been met
- Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation.

➔ **Need of Regression testing**

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fix

22. What is 7 Key principles? Explain in detail?

- Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or 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.

➔ Below are the 7 key principles

- Testing shows the presence of Defects
- Exhaustive testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context dependent
- Absence of errors fallacy

a) Testing shows the present of Defects

- The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

b) Exhaustive testing is not possible

- Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

- Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

c) Early Testing

- Here early testing means that all the testing activities should start in the early stages of the software development life cycle's requirement analysis stage to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.
- To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

d) Defect Clustering

- The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.
- These types of software or the application will follow the Pareto Principle, which states that we can identify that approx. Eighty percent of the

complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

e) Pesticide Paradox

- This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

f) Testing is context dependent

- Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

g) Absence of errors fallacy

- Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.

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

-

Sr. No.	QA(Quality Assurance)	QC(Quality Control)	Tester

1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented(or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the software.
2	Focuses on processes and procedures rather than conducting actual testing on the system	Focuses on actual testing by executing software with intended to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of software test life cycle(STLC).	QC can be considered as the subset of quality assurance.	Testing is the subset of quality control.

24. Difference between smoke and sanity?

-

Sr. No.	Smoke Testing	Sanity Testing
1	It is a broad approach to testing where all parts of the application are tested.	It is a narrow approach to testing where specific parts of the application are tested.

2	It measures the stability of the system by performing rigorous testing.	It measures the rationality of the system by performing rigorous testing.
3	Smoke testing can be either manual or automated.	Sanity testing can be done without test cases or scripts.
4	It is performed by both testers and developers.	It is performed by only testers.
5	Testing is done without getting into deep but whenever needed tester has to go into deep.	Sanity testing does not need to go into deep of the application.
6	Smoke testing is the first testing performed on the initial build.	Sanity testing is performed when the build is comparatively stable.
7	Smoke testing is documented.	Sanity testing is not documented.
8	It is considered as a subset of acceptance testing.	It is considered as a subset of regression testing.

25. Difference between verification and validation

Sr No.	Verification	Validation
1	We check whether we are developing the right product or not.	We check whether the developed product is right.
2	Verification is also known as static testing.	Validation is also known as dynamic testing.

3	Verifications includes different methods like inspections, reviews and walkthroughs.	Validation includes different testing like functional testing, system testing, integration and user acceptance testing.
4	It is a process of checking the work products of a development cycle to decide whether the products meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow specified business requirements.
5	Quality assurance comes under verification testing.	Quality control comes under validation testing.
6	The execution of code does not happen in the verification testing.	In validation testing execution of code happens.
7	Verification is done before the validation testing.	In validation testing , the execution of code happens.

26. Explain types of performance testing.

- Types of the performance testing are:
  - Load Testing
  - Stress Testing
  - Endurance Testing
  - Spike Testing
  - Volume Testing
  - Scalability Testing

a) Load Testing

- Load testing is a type of testing which involves evaluating the performance of the system under the expected workload. A typical load test includes determining the response time, throughput, error rate, etc during the course of the load test.

- Example – For a newly developed application with an anticipated load of around 1000 concurrent users. We will create a load test script and configure it with 1000 virtual users and run it for say 1-hour duration. After the load test completion, we can analyse the test result to determine how the application will behave at the expected peak load.

b) Stress Testing

- Stress testing is a type of performance testing where we evaluate the application's performance at a load much higher than the expected load. Another aspect of the stress testing is to determine the break-point of the application, the point at which the application fails to respond in the correct manner.
- Example – For an application with an anticipated load of 1000 users we will run the test with 1200 users and check if the application is robust enough to not crash.

c) Endurance Testing

- Endurance testing is also known as 'Soak Testing'. It is done to determine if the system can sustain the continuous expected load for a long duration. Issues like memory leakage are found with endurance testing.
- Example – For an application like Income tax filing, the application is used continuously for a very long duration by different users. In this type of application, memory management is very critical. For an application like these, we can run the test for 24 hours to 2 days duration and monitor the memory utilization during the whole test execution.

d) Spike Testing

- In spike testing, we analyse the behaviour of the system on suddenly increasing the number of users. It also involves checking if the application is able to recover after the sudden burst of users.
- Example – For an e-commerce application running an advertisement campaign, the number of users can increase suddenly in a very short duration. Spike testing is done to analyse these types of scenarios.

e) Volume Testing

- The volume testing is performed by feeding the application with a high volume of data. The application can be tested with a large amount of data inserted in the database or by providing a large file to the application for processing. Using volume testing, we can identify the bottleneck in the application with a high volume of data.
- Example – For a newly developed e-commerce application, we can perform volume testing by inserting millions of rows in the database and then carry out the performance test execution.

f) Scalability Testing

- Scalability testing is used to determine if software is effectively handling increasing workloads. This can be determined by gradually adding to the user load or data volume while monitoring system performance. Also, the workload may stay at the same level while resources such as CPUs and memory are changed.

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

-

a) Error

- The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The developers use the term error.

b) Defect

- When the application is not working as per the requirement is knows as defects. It is specified as the aberration from the actual and expected result of the application or software.
- In other words, we can say that the bug announced by the programmer and inside the code is called a Defect**.**

c) Bug

- In software testing, a bug is the informal name of defects, which means that software or application is not working as per the requirement. When we have some coding error, it leads a program to its breakdown, which is known as a bug. The test engineers use the terminology Bug.

d) Failure

- Many defects lead to the software's failure, which means that a loss specifies a fatal issue in software/ application or in its module, which makes the system unresponsive or broken.
- In other words, we can say that if an end-user detects an issue in the product, then that particular issue is called a failure.
- Possibilities are there one defect that might lead to one failure or several failures.

28. Difference between priority and severity

-

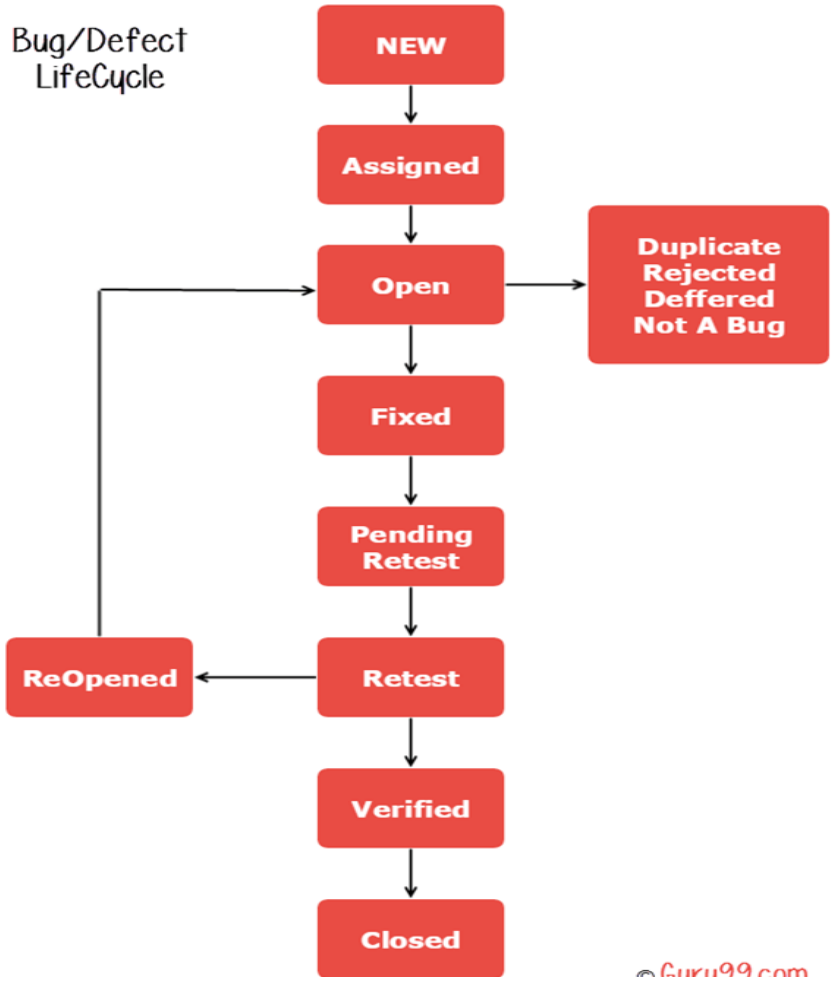
Sr No.	Priority	Severity
1	Defect priority has defined the order in which the developer should resolve a defect	Defect severity is defined as the degree of impact that a defect has on the operation of the product
2	Priority is associated with scheduling.	Severity is associated with functionality or standards
3	Priority indicates how soon the bug should be fixed.	Severity indicates the seriousness of the defect on the product functionality.
4	Priority of defect is decided in consultation with the manager or client.	QA engineer determines the severity level of the defect.
5	Priority is driven by business value.	Severity is driven by functionality.
6	Priority status is based on customer requirements.	Severity status is based on technical aspect of the product
7	Priority categorised into three types - Low - Medium - High	Severity is categorised in five types - Critical - Major - Moderate - Minor

		- Cosmetic

-

29. What is bug life cycle?

- A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program's source code or its design."
- The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as 'Defect Life Cycle'.
- When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.
- The process by which the defect moves through the life cycle is depicted next slide.



➔ **Stages of Defect**

a) New : When a new defect is logged and posted for the first time. It is assigned as status new.
b) Assigned : Once the bug is posted by the tester, the lead of tester approves the bug and assigns the bug to the developer team
c) Open : The developer starts analysing and works on the defect fix
d) Fixed : When a developer makes a necessary code change and verifies the change, he or she can make bug status as fixed.
e) Pending Retest : Once the defect is fixed the developer gives a particular code for retesting the code to tester. since the software testing remains pending from the testers end, the status assigned is Pending retest.
f) 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 status to Retest.
g) Verified : The tester retests the bug after it got fixed by the developer. If there is no bug detected in the software , then bug is fixed and status assigned is verified.
h) Reopen : If the bug persists even after developer has fixed the bug. The tester changes the status to reopened. Once again the bug goes through the life cycle.
i) Closed : If the bug is no longer exists then tester assigns the status closed.
j) Duplicate : If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to duplicate.
k) Rejected : If the developer feels the defect is not a genuine defect then it changes defect to rejected.
l) Deferred : If the present bug is not a prime priority and if it is expected to get fixed in the next release , then status Deferred is assign to such bugs.
m) 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. Explain the difference between functional testing and non-functional testing.

-

Sr. 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 non-functional 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 must be used for functional testing.	Using tools will be effective for the non-functional testing.
4	Business requirements are the inputs to the functional testing.	Performance parameters like speed, scalability are inputs to non-functional testing.
5	Functional testing describes what the product does.	Non-functional testing describes how good the product works.
6	Easy to do manual testing.	Tough to do manual testing.
7	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 - Penetration testing etc. - Migration Testing - Compatibility Testing

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

-

Sr. No.	SDLC	STLC
1	It is primarily connected to software development, which means that it is the procedure of developing a software application.	It is mainly linked to software testing, which means that it is a software testing process that contains various phases of the testing process.
2	SDLC stands for software development life cycle.	STLC stands for software testing life cycle.
3	While performing the SDLC process, we needed a greater number of developers to complete the development process.	The STLC process needed a smaller number of testers to complete the testing process.
4	Besides development phase, other phases like testing are also included.	The STLC only concentrate on testing the software.
5	SDLC will help us to develop a good quality software product.	STLC will helps to create the software bug free.
6	The various phases of SDLC are : - Requirement collection - Feasibility Study - Design - Coding - Testing - Installation - Maintenance	The various phases of STLC are : - Requirement collection - Test Plan - Write test case - Traceability Matrix - Defect Tracking - Test Execution Report - Retrospect meeting
7	The SDLC phases are done before the STLC phases.	The STLC phases are done after SDLC phases.

-

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

-

Sr No.	Test Scenario	Test Case	Test Script
1	Is any functionality that can be tested	Is a set of actions executed to verify particular features or functionality.	Is a set of instructions to test an app automatically.
2	It derives from test artifacts like BRS and SRS	It mostly derive from test scenarios.	It is mostly derived from test cases.
3	Helps test the end to end functionality in an agile way	Helps in exhaustive testing of an app.	Helps to test specific things repeatedly.
4	Is more focus on what to test	Is focused on what to test and how to test	Is focused on the expected result.
5	Take less time and fewer resources to create.	Require more resources and time.	Require less time for testing but more resources for scripts creating and updating.
6	Allows quickly assessing the testing scope.	Allows detecting errors and defects.	Allows carrying out an automatic execution of test cases.

33. Explain what test plan is? What is the information that should be covered.

- A document describing the scope, approach, resources and schedule of intended test activities
- Determining the scope and risks, and identifying the objectives of testing.
- Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
- Integrating and coordinating the testing activities into the software life cycle activities:
    - acquisition, supply, development, operation and maintenance.
- Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, and how the test results will be evaluated?
- Scheduling test analysis and design activities.
- Scheduling test implementation, execution and evaluation.
- Assigning resources for the different activities defined
    - Defining the amount, level of detail, structure and templates for the test documentation.

34. What is priority?

- Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.
- **For example:** If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

➔ **Priority can be of following types:**

a) **Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.
b) **Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
c) **High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.
d) **Critical:** Extremely urgent, resolve immediately

35. What is severity?

- Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.
- **For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

➔ **Severity can be of following types:**

a) **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
b) **Major (High):** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
c) **Moderate (Medium):** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
d) **Minor (Low):** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
e) **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

36. Advantage of Bugzilla.

-

➔ **Advantages of Bugzilla are :**

- It improves the quality of the product
- It enhances the communication between the developing team and testing team.

- It has the capability to adapt multiple situations.
- Time tracking
- Localization
- Customization

37. Difference between priority and severity

-

Sr No.	Priority	Severity
1	Defect priority has defined the order in which the developer should resolve a defect	Defect severity is defined as the degree of impact that a defect has on the operation of the product
2	Priority is associated with scheduling.	Severity is associated with functionality or standards
3	Priority indicates how soon the bug should be fixed.	Severity indicates the seriousness of the defect on the product functionality.
4	Priority of defect is decided in consultation with the manager or client.	QA engineer determines the severity level of the defect.
5	Priority is driven by business value.	Severity is driven by functionality.
6	Priority status is based on customer requirements.	Severity status is based on technical aspect of the product
7	Priority categorised into three types - Low - Medium - High	Severity is categorised in five types - Critical - Major - Moderate - Minor - Cosmetic

38. What are the different methodologies in agile development model?

- The below agile methodologies list comprises famous types of agile methodology that one can opt from:

1) Kanban
2) Scrum
3) Extreme Programming (XP)
4) Crystal
5) Dynamic Systems Development Method (DSDM)
6) Feature-Driven Development (FDD)

39. Explain the difference between authorization and authentication in web testing. What are the common problems faced in web testing?

-

Sr No.	Authentication	Authorization
1	Authentication is the process of identifying user to provide access to a system.	Authorization is the process of giving permission to access the resources.
2	In this user or client and server are verified.	In this it verify that user is allowed through the defined policies and rules.
3	It is usually perform before the authorization.	It is usually done once the user is successfully authenticated.
4	It requires login details of user, such as user name and password	It requires user privileges or security level.
5	Data is provided through token ids.	Data is provided through access tokens.

6	Authentication credentials can be partially changed by the user as per the requirement.	Authorization permissions can not be changed by the user. The permissions are given to a user by the owner or manager of the system , and he can only change it.

➔ **Common problem faced in web testing**

a) Cross browser Compatibility
b) Responsiveness
c) Cross device compatibility
d) Integration Testing
e) Security
f) Performance Testing
g) Application Getting Slow
h) Usability Testing
i) Entry and Exit Points
j) Checking the standards and compliance
k) Firewalls
l) Accessibility Testing
m) Project Deadline
n) User Experience
o) Web Service Request
p) User Input Validation

40. When to use usability testing?

- If possible, usability testing can and should be conducted on the current iteration of a product before beginning any new design work, after you've begun the strategy work around a brand new site or app. This will quickly identify areas for opportunity, and reduce the amount of assumptions your design team will make with regard to what the user wants. Additionally, after the usability tests analysis, the team should have the ability to pinpoint the steps needed to achieve the project goals with as little disruption as possible.
- Don't assume that a system is completely broken when beginning a project. Most likely designers, developers, researchers, content strategists, etc. have already spent a lot of time building what you see

before you. Rather than assuming that the efforts of previous teams were completely misguided, identify particular areas where design, testing, and validation can be conducted in order to enhance and correct the product. Ultimately, this will assist in limiting the scope of work.

- Once you've gotten results from an initial usability test, it's then important to use those results throughout your design phase and keep re-testing users. We typically design at the wireframe level first, followed by the high-fidelity final designs. At both of those stages, we create clickable prototypes using InVision, which allow us to perform user tests and continue to optimize the design and usability of the site.
- Given the amount of usability testing that occurs in our process, we always launch our sites with the knowledge that the new site will significantly outperform the old one. The data backs it up. Once a new site is launched, we continue to perform periodic user tests in order to improve the site consistently. Remember, any website or app is a living, breathing, digital animal – it must be well taken care of to grow into a fully mature platform, and usability tests are a significant tool to get there.

41. What is the procedure of GUI Testing?

- Graphical User Interface (GUI) testing is the process of testing the system's GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars - tool bar, menu bar, dialog boxes and windows etc.

➔ **What do you check in GUI Testing?**

- Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in application is readable
- Check the alignment of the text is proper
- Check the colour of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

➔ **Approach of GUI testing**

- Manual based testing
- Record and replay
- Model based testing

➔ **<u>Example</u>**

- Web based testing & desktop based testing
- Mobile based testing
- Game based testing