MODULE 2 [MANUAL TESTING]

## Q1 : What is Exploratory Testing ?

ANS :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.

. This testing is useful when requirement documents are not available or partially available.

# Q2 : What is traceability matrix ?

ANS : There are three types of traceability matrix.

. A requirements traceability matrix (RTM) is a little more complex and involves tracking the business requirements against the functional requirements.

.This requires the mapping of technical needs of a project against the business requirements.

. Forward traceability which is used to check if the project progresses in the right direction for the right product

. It helps uncover if the requirements are applied to the product.

. Second is backward or reverse traceability that helps to assess whether the product is on track.

. This helps to keep the project from suffering from scope drift.

. Finally, bi-directional traceability ensures that all requirements are covered by test cases and then helps you to analyse the impact of changes to requirements .

### Q3 : What is Boundary value testing ?

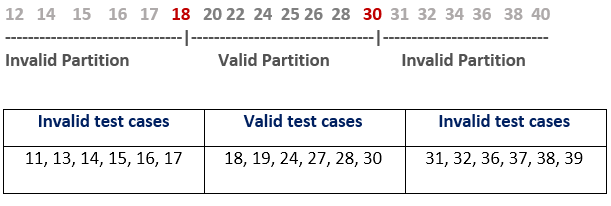
ANS : Boundary value analysis is one of the widely used case design technique for black box testing.

.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.

FOR EXAMPLE : Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



# Q4 : What is Equivalence partitioning testing ?

ANS : The Equivalence Partitioning Testing technique is one of the popular software testing techniques.

.It is used to reduce the number of test cases by creating boundaries that ensure code paths are not executed for specific input data values.

. [Equivalence Partitioning](https://www.softwaretestingo.com/equivalence-partitioning-testing/), helps you divide input data into several partitions based on the characteristics of the inputs, each having the same set of boundary values for one variable, but all having different sets of boundary values for another variable.

. In other words, each partition is a collection of inputs with the same value(s) for one variable and different values for some other variable.

. Creating partitions by splitting every set of input values into two sub-sets so each partition contains input values with different boundaries for one variable but the same boundary for another variable.

**Q5 : What is Integration testing ?**

ANS : Integration testing is known as the second level of the [software testing process](https://www.simplilearn.com/tutorials/devops-tutorial/fundamentals-of-software-testing), following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed.

.As a rule, the usual software project consists of numerous software modules, many of them built by different[programmers.](https://www.simplilearn.com/how-to-become-programmer-article) Integration testing shows the team how well these disparate elements work together. After all, each unit may function perfectly on its own, but the pressing question is, “But can they be brought together and work smoothly.

. So, integration testing is the way we find out if the various parts of a software application can play well with others.

**Q6:What determines the level of risk?**

**Q7:What is Alpha testing?**

**Q8:What is beta testing?**

Q9: What is component testing?

**ANS .** Component testing is a type of white box testing where you validate an individual component of the application before testing the entire application.

. Component testing finds bugs and verifies the functionality of software modules/programs which are individually testable.

.Component testing is also known as program or module testing.

.It results in the detection of defects in software modules and checks the functioning of the software.

. Testing each component separately is termed as component testing.

# Q10 : What is functional system testing?

ANS . Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements.

.Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user’s expectations.

.The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

. Some functional testing examples are

. Can users successfully log in to the application once they provide legitimate credentials?

.Does the payment gateway reject the input and display an error message when a user keys in an invalid credit card number?

.Do inputs to the “Add New Record” screen successfully add and save a new record to the database?

**Unit Testing**: This is performed by developers who write scripts that test if individual components/units of an application match the requirements. This usually involves writing tests that call the methods in each unit and validate them when they return values that match the requirements.

**Smoke Testing**: This is done after the release of each build to ensure that software stability is intact and not facing any anomalies.

**Sanity Testing**: Usually done after smoke testing, this is run to verify that every major functionality of an application is working perfectly, both by itself and in combination with other elements.

**Regressing testing** .This test ensures that changes to the codebase (new code, debugging strategies, etc.) do not disrupt the already existing functions or trigger some instability.

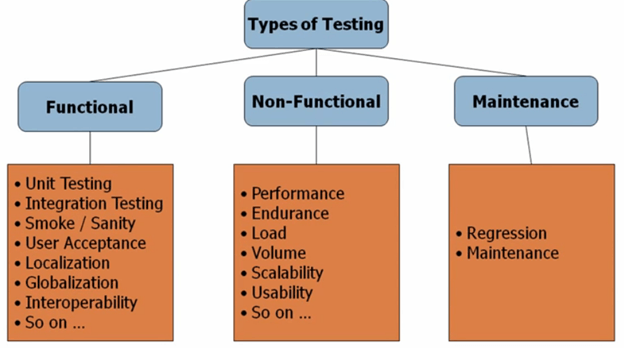
**Integration Testing**: If a system requires multiple functional modules to work effectively, integration testing is done to ensure that individual modules work as expected when operating in combination with each other. It validates that the end-to-end outcome of the system meets these necessary standards.

## Q11. What is Non-Functional Testing?

ANS: **Non-Functional Testing** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

.An excellent example of non-functional test would be to check how many people can simultaneously login into a software .

.Non-functional testing is equally important as functional testing and affects client satisfaction.



Q12.What is GUI Testing?

ANS . GUI testing is described as the testing of the application under the Test system graphical user interface.

.Graphical Testing consists of screen checks using controls such as icons, buttons, and menus with all types of the bar such as dialogues, menu bar, toolbar, Windows, etc. Basically, we can say that what we see is GUI.

.For a computer application, there are two kinds of interfaces.

.The command-line interface is the one that answers to the command text and the computer.

. On the other hand, [GUI](https://www.educba.com/what-is-gui/) is a graphical user interface in which you use pictures rather than text to communicate with the computer.

# Q13 . What is adhoc testing ?

# Q14. What is load testing ?

ANS. Load Testing is a type of**performance testing** in which any software, application, or website is subjected to a realistic load.

. Your application could fail badly in real-world scenarios if you don't have it.

.When your software development project is nearing completion, there’s one test that’s essential to understanding its readiness for deployment: load testing.

.  During a load test, testing software will measure the capacity of your web application via transaction response times.

. Load testing helps developers identify issues like system lag, slow page load times, or crashes when different levels of traffic are accessing the application during production rather than post-launch.

# Q15 : What is stress Testing ?

ANS : Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress.

The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

. It allows the test team to monitor system performance during failures.

.To verify if the system has saved the data before crashing or NOT.

.To verify if the system prints meaning error messages while crashing or did it print some random exceptions.

.To verify if unexpected failures do not cause security issues.

# Q16 : What is white box testing and list the types of white box testing ?

ANS: White Box Testing is a software examining technique that involves testing the product's underlying structure, design, and coding in order to verify input-output flow and improve design, usability, and security

. White box testing is also known as Clear box testing, Open box testing, transparent box testing, Code-based testing, and Glass box testing since the code is visible to the testers.

## . Example of White Box Testing

**Code −**

Printme (int a, int b) {------------ it is a function

int result = a+ b;

If (result > 0)

Print ("Positive", result)

Else

Print ("Negative", result)

} ----------- End of the code

. Types Of White box testing

**1.Unit Testing**:

Unit testing is a fairly straightforward white box test. It tests the structure of a program at the smallest level. The idea is to check whether individual sections of source code are working per design or not. It can also be used to see if specific points in the code are reached or whether any lines are executed during program execution.

For example, a developer has written a C# method called ComputeTax(). A unit test would call this method with different parameters to see if the result is correct each time.

**2. Smoke testing**

Smoke testing is an initial white box testing that may be performed when delivering a large project or releasing software for internal use. The purpose of smoke testing is to ensure basic functionality. This involves a cursory check to see if the “heart” of a program is working. This usually involves black box testing, but it doesn’t have to be.

For example, you may run a smoke test on a newly developed software program to ensure that the program’s main functions work as expected.

**3. Sanity Testing**

Sanity testing is a more in-depth type of white box testing used to verify a system’s basic functionality after major changes, such as code refactoring or updates to the database. To sanity test a system, you would run all the unit tests and smoke tests that you normally would, but you would also test specific use cases known to break the system.

For example, after making some changes to a web application’s code, you might sanity test the application by loading it in a browser and performing some basic tasks, like entering data into form fields and clicking on buttons.

**4. Code coverage testing**

 Code coverage testing is used to measure the degree to which the source code of a program is tested. This type of white-box testing is usually done by tools that “trace” the program’s execution flow and generate reports on what lines of code were executed.

For example, you can use code coverage tools to determine what percentage of your source code is being executed during the [testing of a program](https://www.hikeqa.com/mobile-app-testing-services/manual-testing-v-s-automated-testing/). This can help spot code sections that are not essential to the program’s success or are difficult to maintain.

**5. Data flow testing**

 Data flow testing is used to determine how data flows through an application by checking individual paths through all components in an application. It is performed by tracing the path of data from input to output.

For example, you can use data flow testing to see if certain inputs result in the expected outputs. This can help you to identify errors in your application.

**6. Path testing**

Path testing is a type of data flow testing that checks the flow through an application along a particular path.  Testing aims to ensure that data travels along expected paths through an application, regardless of its starting point or the inputs used to produce outputs.

For example, one might perform path testing by entering various incorrect inputs into an application and comparing these with the results obtained when using the correct inputs. This would help identify any differences between the expected outputs and those that are reproduced.

**7. Statement testing**

Statement testing is a type of data flow testing that checks whether individual lines of source code perform as expected or not. This type of white box testing verifies the behavior of each line of code in an application.

For example, you can use statement testing to see if a particular line of code is executed when a particular condition is met. This can help point out problems with the code.

## Q17 : What is black box testing? What are the different black box testing techniques?

ANS : Black box testing techniques apply to all levels of testing, as well as functional and non-functional testing types.

.There are four main black box testing techniques: equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.

. The test cases which are designed to test a system plays an important role in testing. The way they are created and the scenarios they cover should be taken into consideration.

Testers can create requirement specification document by using the below techniques:

[Equivalence Partitioning](https://www.educba.com/equivalence-partitioning/)

Boundary Value Analysis

Decision Table Testing

State Transition Testing

Error Guessing

Graph-based Testing Methods

Comparison Testing

Use Case Technique

Following are the Techniques explained below:

1. Equivalence Testing

This technique divides the input values which are provided to the software into different groups or classes. This is done on the basis of the output which will be coming as an outcome. This technique is also known as Equivalence Class Partitioning. By doing this we save the effort of giving different inputs. Instead, we give one value to the group or class to test the outcome for that group or class. This helps in improving the test coverage and in turn reducing the rework. The time is also saved as no separate inputs are to be given. Input for each class is sufficient.

Let us take an example of marks that students score. If a student scores above75% then he/she has secured First class with Distinction. Similarly, if the score is between 60% to 75% then he/she has secured First Class. If the score is between 50% to 60% then Second Class. If the score is between 40% to 50% then Pass class, else fail. Here there will be four classes. These test cases are formed and it is made sure that all possibilities are hence covered. Hence testing with any values in this set is sufficient.

2. Boundary Value Analysis

Here the focus is on the values which are present at the boundaries. This is because usually there are many issues found when it comes to testing with values that focus on boundaries. Boundary focuses on values near the limit where the behaviour of the system changes. In boundary value analysis both inputs, which are valid and invalid are to be tested.

For example, if we want to test values that range from 1 to 100 then we should check if how the program works for values like 1-1, 1+1, 1, 100-1, 100+1, etc. This helps in saving time again as we can only check for values like 0, 1, 2, 99, 100 and 101.

3. Decision Table Testing

Whenever there are logical conditions or decision-making steps then this technique is to be used. These can be like if a particular condition is not satisfied then Action A should be performed, else Action B is to be performed. The tester needs to identify the input and actions which are to be performed based on the conditions. A decision table is created based on these. Consider an example where an odd number of vehicles are allowed only on Monday, Wednesday, Friday and Sunday, while even a number of vehicles are allowed on Tuesday, Thursday and Saturday. In this case, there are two conditions and two actions. Condition 1 being odd vehicles and Condition 2 being even vehicles. The actions being the days when these vehicles can be on the roads. The total number of test cases, in this case, can be four and hence the decision table can be derived accordingly.

4. State Transition Testing

In this technique, the test case tries to test the system under different states. This state can change depending upon different conditions or events. When a particular event occurs then these scenarios can be tested.

5. Error Guessing

This technique is mainly based on experience. Once a tester has experience working on any application its behavior and functionalities are known to him/her. This is a way through which many issues can be found out. By using this experience, it is easy for the testers to guess where most developers are prone to make mistakes. These can be handling null values, accepting the submit button without any value, uploading a file without any attachment, uploading a file with less than or more than the limit size specified, etc.

6. Graph-based Testing

Each application is built by using some objects. All the objects which are used are noted and a graph is prepared. From this graph, the relationship of every object is identified, and test cases are written accordingly.

7. Comparison Testing

In this technique, [different versions](https://www.educba.com/tableau-versions/) of the same software are used and then compared in order to test the entire system. The behaviour is noted and compared for all versions and any deviations are noted.

8. Use Case Technique

This technique is used to identify all the [test cases in use](https://www.educba.com/test-cases-vs-test-scenario/) as per the system. All scenarios are noted which help in understanding the complete functionality of each function in an end to end way. The test cases should have cases that cover all scenarios from beginning to end as per the system usage.

# Q18 : Mention what are the categories of defects?

ANS : 1) Wrong: If the requirements are implemented incorrectly, then they are stated as Wrong defects.

2) Missing: If the requirement is not done which is given by the customer. ...

3) Extra: If a requirement is not given by the end user and if it is done,then it is called as an extra defect.

# Q19 : Mention what bigbang testing is?

ANS: Big Bang Integration Testing is an integration testing strategy, wherein all units are linked at once, which results in a complete and efficient system.

. In this type of integration testing all the components as well as the modules of the software are integrated simultaneously, after which everything is tested as a whole.

.During the process of big bang integration testing, most of the developed modules are coupled together to form a complete software system or a major part of the system, which is then used for integration testing.

.This approach of software testing is very effective as it enables software testers to save time as well as their efforts during the integration testing process.

# Q20 : What is the purpose of exit criteria?

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

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

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.

**Q21.When should "Regression Testing" be performed?**

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

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

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

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

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

**Q22.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. Here, in this section, we are going to learn about the seven essential principles of software testing.

seven different testing principles

1.Testing shows 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

**1.Testing shows presence of defects**

Hence, testing principle states that – Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

**2.Exhaustive testing is not possible**

Exhaustive testing is not possible. Because of Instead, we need the optimal amount of testing based on the risk assessment of the application.

And the million dollar question is, how do you determine this risk?

To answer this let’s do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle defect Clustering

**3.Early testing**

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial

**4.Defect clustering**

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

**5.Pesticide paradox**

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

**6.Testing is context dependent**

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software’s are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine

**7.Absence of errors fallacy**

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user’s needs & requirements.

**Q23.Difference between QA v/s QC v/s Tester**

**QA QC TESTER**

|  |  |  |
| --- | --- | --- |
| QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements. | It includes activities that ensure the identification of bugs/error/defects in a software. |
| Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing the software with an aim to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| Process-oriented activities. | Product-oriented activities. | Product-oriented activities. |
| Preventive activities. | It is a corrective process. | It is a preventive process. |
| 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. |

**Q24.Difference between Smoke and Sanity?**

**Smoke testing Sanity testing**

|  |  |
| --- | --- |
| The goal of this testing is to confirm that the recently made build is steady enough to perform further rigorous testing and can the basic functionalities are working properly. | The goal is to test whether the bugs are fixed after the stable build and determine the system’s correctness. It just includes the module on which code changes take place. |
| This testing can be performed by two ways- manually and automatically. | This testing can be done without any test cases and scripts. |
| The software can be stable or unstable in smoke testing. This testing is done after releasing every new build, including an end-to-end system verification. So, in-depth testing is necessary for this. | For this test, the software should be comparatively stable. It is done for specific component verification, like a newly introduced feature. So, in-depth testing is not possible for this. |
| Executing smoke testing helps to ensure that the issues fixed on the previous build are not bothering the vital functionalities of the application. | Executing sanity testing helps to save unnecessary testing effort and time because it’s only performed on some specific functionalities. |
| Test documents and scripts are made for future reference. | There is no test document or test script needed for this testing. So, no future reference will be available. |
| It is assumed as a subset of acceptance testing. | Assumed as a subset of regression testing |

**Q25.Difference between verification and Validation**

**Verification validation**

|  |  |
| --- | --- |
| It is the static practice of studying and verifying the specific requirements of a particular stage in development. | It is the dynamic practice of testing the final product after development to ensure it meets customer requirements. |
| It does not require executing code. | It always requires executing code. |
| This involves only human verification of required assets. | This involves both human and machine-based checking and approval of software. |
| It uses document reviews, inspections, product walkthroughs, and desk-checking techniques. | It involves various types of product testing – unit tests, integration tests, regression tests, cross-browser and cross-device testing, etc. |
| It is meant to detect bugs at the beginning of each development phase. | It is meant to detect all unnoticed bugs at the verification stage. |
| Its targets are specification documents, application and software architecture design docs, ER diagrams, database table design, test cases and scenarios, traceability matrix, etc. | Its target is the product to be used by the customer after public release. |
| It is undertaken by both developers and testers to ensure that the software adheres to predetermined standards and exactions. | It is primarily undertaken by experienced Quality Assurance engineers who comb through all features of an application to ensure they work as expected. |
| It comes before validation testing. | It follows verification testing. |
| It does not require any devices, platforms, browsers, or operating systems for its execution. | It is best executed by using real browsers, devices, and operating systems. |

**Q26.Explain types of Performance testing.**

**Load testing**

The load testing is used to check the performance of an application by applying some load which is either less than or equal to the desired load is known as load testing.

**Stress Testing**

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load.

**Stability Testing**

Checking the performance of an application by **applying the load for a particular duration of time** is known as **Stability Testing**

**Scalability Testing**

Checking the performance of an application by increasing or decreasing the load in particular scales (no of a user) is known as **scalability testing**. Upward scalability and downward scalability testing are called scalability testing

**Q27.What is Error, Defect, Bug and failure?**

**Error:**

A discrepancy between a computed, observed, or measured value or condition and

the true, specified, or theoretically correct value or condition. This can be a

misunderstanding of the internal state of the software, an oversight in terms of memory

management, confusion about the proper way to calculate a value, etc.

**Defect:**

Commonly refers to several troubles with the software products, with its external

behavior or with its internal features.

**Bug**:

A fault in a program which causes the program to perform in an unintended or

unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology

of Tester.

**Failure:**

The inability of a system or component to perform its required functions within

specified performance requirements. See: bug, crash, exception, and fault.

**Q28.Difference between Priority and Severity**

| **Priority** | **Severity** |
| --- | --- |
| 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 |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |
| Priority is categorized into three types  Low  Medium  High | Severity is categorized into five types  Critical  Major  Moderate  Minor  Cosmetic |

**Q29.What is Bug Life Cycle?**

****

As you can see from above diagram, a defect‘s state can be divided into Open or Closed.

When a bug reaches one of the Closed or Terminal states, its lifecycle ends. Each state has

one or more valid states to move to.

This is to ensure that all necessary steps are taken to resolve or investigate that defect. For

example, a bug should not move from Submitted state to resolved state without having it open.

In a typical scenario, as soon as a bug is identified, it is logged into the bug tracking system

with status as Submitted. After ascertaining the validity of the defect, it is given the “Open”

Status.

**Q30.Explain the difference between Functional testing and NonFunctional testing**

|  |  |
| --- | --- |
| Functional testing | Non-functional testing |
| It focuses on testing the functionality of the software or system. | It focuses on testing the non-functional aspects of the software or system. |
| Verifies whether the software meets the functional requirements. | Verifies whether the software meets the non-functional requirements such as performance, security, usability, reliability, and compatibility. |
| It involves testing the features and functionalities of the software, such as input/output, error handling, and user interface. | It involves testing the quality attributes of the software, such as response time, scalability, availability, and maintainability. |
| Tests are typically conducted using test cases or scenarios that validate the functional requirements. | Tests are conducted using various techniques such as load testing, stress testing, security testing, and usability testing. |
| It can be performed manually or using automated testing tools. | Often requires specialized testing tools and frameworks to measure and evaluate the non-functional requirements. |
| Done after unit testing and integration testing and before system testing. | It can be done at various stages of the development lifecycle, from design to deployment and maintenance. |

**Q31.What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle**)?

| SDLC | STLC |
| --- | --- |
| SDLC is mainly related to software development. | STLC is mainly related to software testing. |
| Besides development other phases like testing is also included. | It focuses only on testing the software. |
| SDLC involves total six phases or steps. | STLC involves only five phases or steps. |
| In SDLC, more number of members (developers) are required for the whole process. | In STLC, less number of members (testers) are needed. |
| In SDLC, development team makes the plans and designs based on the requirements. | In STLC, testing team(Test Lead or Test Architect) makes the plans and designs. |
| Goal of SDLC is to complete successful development of software. | Goal of STLC is to complete successful testing of software. |
| It helps in developing good quality software. | It helps in making the software defects free. |
| SDLC phases are completed before the STLC phases. | STLC phases are performed after SDLC phases. |
| Post deployment support , enhancement , and update are to be included if necessary. | Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts. |
| Creation of reusable software systems is the end result of SDLC. | A tested software system is the end result of STLC |

**Q32.Explain what Test Plan is? What is the information that should be covered.**

A Test plan is a document that consists of all future testing-related activities. It is prepared at the project level and in general, it defines work products to be tested, how they will be tested, and test type distribution among the testers.

**Overview of testing activities:** The test plan provides an overview of the testing activities and where to start and stop the work.

**Provides timeline:**The test plan helps to create the timeline for the testing activities based on the number of hours and the workers needed.

**Helps to estimate resources:** The test plan helps to create an estimate of the number of resources needed to finish the work.

**Serves as a blueprint:**The test plan serves as a blueprint for all the testing activities, it has every detail from beginning to end.

**Helps to identify solutions:**A test plan helps the team members They consider the project’s challenges and identify the solutions.

**Serves as a rulebook:**The test plan serves as a rulebook for following rules when the project is completed phase by phase.

Q33.What is priority?

Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.

**Types of Priority**

**Low:**The Defect is an irritant but repair can be done once the more serious Defect has been fixed

**Medium:**During the normal course of the development activities defect should be resolved. It can wait until a new version is created

**High:**The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed

Q34.What is severity?

Priority is important for fixing the bug or which bug to be fixed first or how soon the bug should be fixed.

**High:** it is a major impact on the customer application, and it has to be fixed first.

**Medium:** In this, the problem should be fixed before the release of the current version in development.

**Low:** The flow should be fixed if there is time, but it can be deferred with the next release.

**Q35.Bug categories are…**

**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 bugs 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. Function 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 requriments 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 analyze 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

**Q36.Advantage of Bugzila .**

It improves the quality of the product.

It enhances the communication between the developing team and the testing team.

It has the capability to adapt to multiple situations.

Bugzilla is extremely good at monitoring and handling bugs and issues.

You can use the ‘request system’ provided by Bugzilla to ask other users to evaluate codes, provide information and other things

**Q37.Difference between priority and severity**

| **Priority** | **Severity** |
| --- | --- |
| 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 |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |
| Priority is categorized into three types  Low  Medium  High | Severity is categorized into five types  Critical  Major  Moderate  Minor  Cosmetic |

**Q38.What are the different Methodologies in Agile Development Model?**

**Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

**Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

**Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

**Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

**Deployment:** In this phase, the team issues a product for the user's work environment.

**Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

**Q39.Explain the difference between Authorization and Authentication in Web testing.What are the common problems faced in Web testing?**

| **Authentication** | **Authorization** |
| --- | --- |
| In the Authentication process, the identity of users are checked for providing the access to the system. | While in Authorization process, a the person’s or user’s authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user’s login details. | While it needs the user’s privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines **What permission does the user have?** |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process. | The OAuth 2.0 protocol governs the overall system of user authorization process. |
| Popular Authentication Techniques-  Password-Based Authentication  Passwordless Authentication  2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication)  Social authentication | Popular  Authorization Techniques-  Role-Based Access Controls (RBAC)  SAML Authorization  OpenID Authorization  OAuth 2.0 Authorization |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. |
| **Example**: Employees in a company are required to authenticate through the network before accessing their company email. | **Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access. |

**Q40.When to used Usablity Testing?**

When software is ready, it is important to make sure that the user experience with the product should be seamless. It should be easy to navigate and all the functions should be working properly, the competitor’s website will win the race. Therefore, usability testing is performed. The objective of usability testing is to understand customers’ needs and requirements and also how users interact with the product (software). With the test, all the features, functions, and purposes of the software are checked.

The primary goals of usability testing are – discovering problems (hidden issues) and opportunities, comparing benchmarks, and comparison against other websites***.*** The parameters tested during usability testing areefficiency, effectiveness, and satisfaction***.*** It should be performed before any new design is made. This test should be iterated unless all the necessary changes have been made. Improving the site consistently by performing usability testing enhances its performance which in return makes it the best website.

**Q41.What is the procedure for GUI Testing?**

MANUAL BASED TESTING

Under this approach, graphical screens are checked manually by testers in conformance with

the requirements stated in business requirements document.

RECORD AND REPLAY

GUI testing can be done using automation tools. This is done in 2 parts. During Record , test

steps are captured into the automation tool. During playback, the recorded test steps are

executed on the Application under Test. Example of such tools - QTP.

MODEL BASED TESTING

A model is a graphical description of system’s behavior. It helps us to understand and

predict the system behavior. Models help in a generation of efficient test cases using the

system requirements.