ASSIGNMENT – MODULE - 2

1 > What is Exploratory Testing?

- When the tester has not any kind of require any document and information related to the system.
- Then based on knowledge and experience the explore known as exploratory testing.

2 > What is traceability matrix?

- It is a graph of requirement & component that help you to keep the virtual table up to date regarding update.

3 > What is Boundary value testing?

- Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near **the limits of valid ranges.**

- Boundary value analysis is a method which **refines** equivalence partitioning.

4> What is Equivalence partitioning testing?

Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.
In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.

5 >What is Integration testing?

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**.

6>What determines the level of risk?

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.

7>What is Alpha testing?

Alpha testing is the first end-to-end testing of a product to ensure it meets the business requirements and functions correctly. It is typically performed by internal employees and conducted in a lab/stage environment. An alpha test ensures the product really works and does everything it's supposed to do.

??What is beta testing?

Beta testing is an opportunity for real users to use a product in a production environment to uncover any bugs or issues before a general release. Beta testing is the final round of testing before releasing a product to a wide audience.

??What is component testing?

Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It's also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.

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

??What is functional system testing?

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

Types of Functional testing are:

· Unit Testing
· Smoke Testing

· Sanity Testing
· Integration Testing
· White box testing
· Black Box testing
· User Acceptance testing
· Regression Testing

• What is Non-Functional Testing?

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.

Types of Nonfunctional testing are:
· Performance Testing
· Load Testing
· Volume Testing
· Stress Testing
· Security Testing
· Installation Testing
· Penetration Testing
· Compatibility Testing

• What is GUI Testing?

Graphical User Interface tasting of responsiveness of control like menu , buttons , icon , tool bar , dialog box etc .

Types of GUI testing

1) Manual based
2) Record and replay based
3)Model based

• What is Adhoc testing?

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

• What is load testing?

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of

Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

- What is stress Testing?

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

Stress Testing is also known as Endurance Testing.
A most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks**.

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

White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

Following are important WhiteBox Testing Techniques:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage

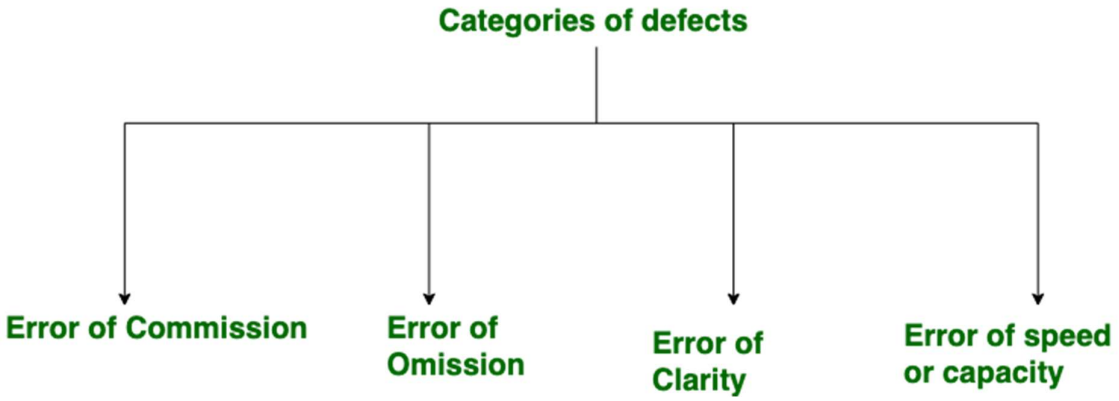
What is black box testing? What are the different black box testing techniques?

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

Following types of black box testing technique:

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

- Mention what are the categories of defects?



- Mention what bigbang testing is?

Big Bang Testing is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

- 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.

- When should "Regression Testing" be performed?

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine.

This testing is done to ensure that new code changes do not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

- What is 7 key principles? Explain in detail?

7 Principles of Software Testing

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) Exhaustive testing is not possible

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

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.

2) 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.

3) 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 defect.

4) Testing shows a 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.

5) Absence of Error – 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.

To solve this problem, the next principle of testing states that Early Testing

6) 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.

7) 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 mach.

Initially, while you learn to drive, you pay attention to each and everything like gear shifts, speed, clutch handling, etc. But with experience, you just focus on driving the rest comes naturally. Such that you even hold conversations with other passengers in the car.

Same is true for testing principles. Experienced testers have internalized these principles to a level that they apply them even without thinking.

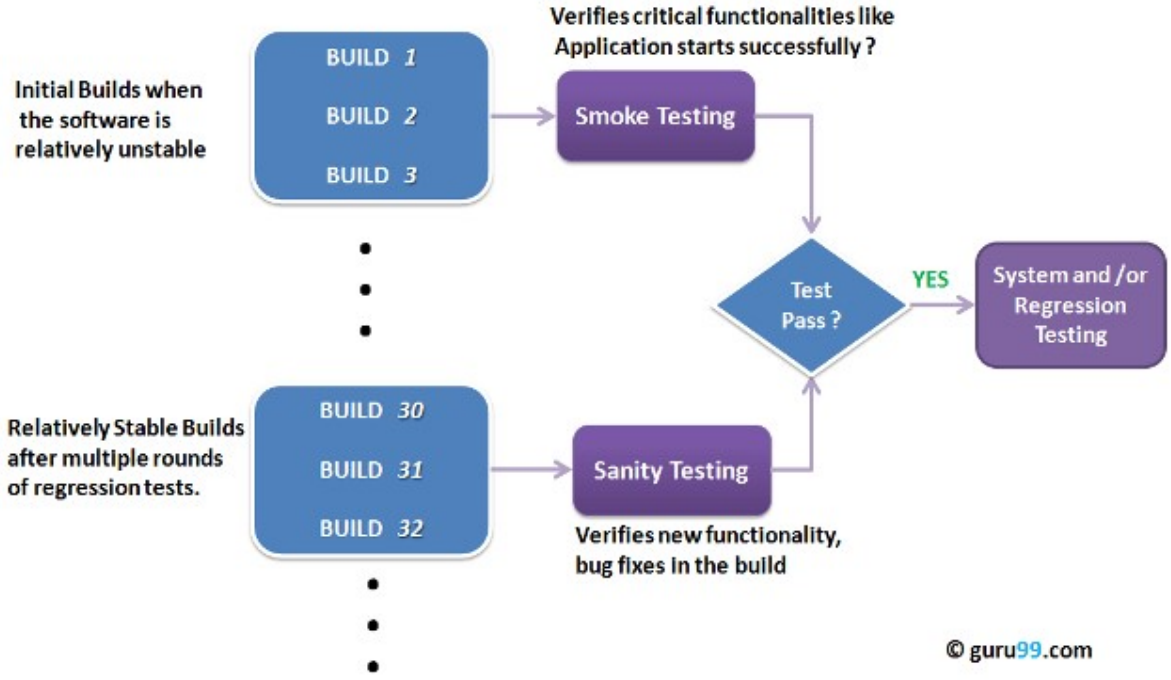
- Difference between QA v/s QC v/s Tester ?

Quality Assurance (QA)	**Quality Control (QC)**	**Testing**
Process-oriented focuses on making the process of creating software better.	**A product-oriented approach is a way to make sure the software meets all its requirements.**	**Testing the software system is about finding any mistakes or issues.**
It works with the development process to help stop mistakes and ensure the software is of good quality.This means setting up and keeping standards, processes, procedures, and tools in place to ensure we're consistently producing high-quality software.	**It's done after the development process and involves running test cases and seeing how the software reacts.**	**This usually happens after the software has been created, and it's all about ensuring that the software's quality is up to standard.**

The goal is to keep improving our software development process for the best possible results.	**The goal is to find any defects or errors in the software and fix them.**	**It involves running tests and looking at what comes out of them, finding any problems with the software, and ensuring that it does everything it's supposed to do.**

- Difference between Smoke and Sanity?

- Smoke Testing has a goal to verify "stability" whereas Sanity Testing has a goal to verify "rationality".
- Smoke Testing is done by both developers or testers whereas Sanity Testing is done by testers.
- Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
- Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
- Smoke testing is documented or scripted whereas Sanity testing isn't.
- Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component.



- Difference between verification and Validation ?

- Verification process includes checking documents, design, code, and program, whereas Validation process includes testing and validation of the actual product.
- Verification does not involve code execution, while Validation involves code execution.
- Verification uses methods like reviews, walkthroughs, inspections, and desk-checking, whereas Validation uses methods like black box testing, white box testing, and non-functional testing.
- Verification checks whether the software confirms a specification, whereas Validation checks whether the software meets the requirements and expectations.
- Verification finds the bugs early in the development cycle, whereas Validation finds the bugs that verification can not catch.
- Comparing validation and verification in software testing, the Verification process targets software architecture, design, database, etc., while the Validation process targets the actual software product.
- Verification is done by the QA team, while Validation is done by the involvement of the testing team with the QA team.
- Comparing Verification vs Validation testing, the Verification process comes before validation, whereas the Validation process comes after verification.

- Explain types of Performance testing.

Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and is also known as ***"Perf Testing"***.

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

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

- What is Error, Defect, Bug and failure?

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

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

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

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

- Difference between Priority and Severity ?

- Priority is the order in which the developer should resolve a defect whereas Severity is the degree of impact that a defect has on the operation of the product.
- Priority is categorized into three types: low, medium and high whereas Severity is categorized into five types: critical, major, moderate, minor and cosmetic.
- Priority is associated with scheduling while Severity is associated with functionality or standards.
- Priority indicates how soon the bug should be fixed whereas Severity indicates the seriousness of the defect on the product functionality.
- Priority of defects is decided in consultation with the manager/client while Severity levels of the defects are determined by the QA engineer.
- Priority value is subjective and can change over a period of time depending on the change in the project situation whereas Severity 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 while High Severity and low priority status indicates defect have to be fixed but not on immediate bases.
- Priority status is based on customer requirements whereas Severity status is based on the technical aspect of the product.

- What is Bug Life Cycle?

Defect Life Cycle or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

- Explain the difference between Functional testing and NonFunctional testing ?

Functional Testing	Non-functional Testing
It verifies the operations and actions of an application.	It verifies the behavior of an application.
It is based on requirements of customer.	It is based on expectations of customer.

Functional Testing	Non-functional Testing
It helps to enhance the behavior of the application.	It helps to improve the performance of the application.
Functional testing is easy to execute manually.	It is hard to execute non-functional testing manually.
It tests what the product does.	It describes how the product does.
Functional testing is based on the business requirement.	Non-functional testing is based on the performance requirement.
Examples: **1.** Unit Testing **2.** Smoke Testing **3.** Integration Testing **4.** Regression Testing	**Examples:** **1.** Performance Testing **2.** Load Testing **3.** Stress Testing **4.** Scalability Testing

- 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.

SDLC	STLC
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.

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

Test Scenario	Test Case	Test Script
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.
Is derived from test artifacts like Business Requirement Specification (BRS) and Software Requirement Specification (SRS).	Is mostly derived from test scenarios.	Is mostly derived from test cases.
Helps test the end-to-end functionality in an Agile way.	Helps in exhaustive testing of an app.	Helps to test specific things repeatedly.
Is more focused on what to test.	Is focused on what to test and how to test.	Is focused on the expected result.
Takes less time and fewer resources to create.	Requires more resources and time.	Requires less time for testing but more resources for scripts creating and updating.
Includes an end-to-end functionality to be tested.	Includes test steps, data, expected results for testing.	Includes different commands to develop a script.
The main task is to check the full functionality of a software application.	The main task is to verify compliance with the applicable standards, guidelines, and customer requirements.	The main task is to verify that nothing is skipped, and the results are true as the desired testing plan.
Allows quickly assessing the testing scope.	Allows detecting errors and defects.	Allows carrying out an automatic execution of test cases.

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

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

- Bug categories are…?

7 Common Types of Software Bugs Every Tester Should Know

- Functional Bugs.
- Logical Bugs.
- Workflow Bugs.
- Unit Level Bugs.
- System-Level Integration Bugs.
- Out of Bound Bugs.
- Security Bugs.

- Advantage of Bugzila?

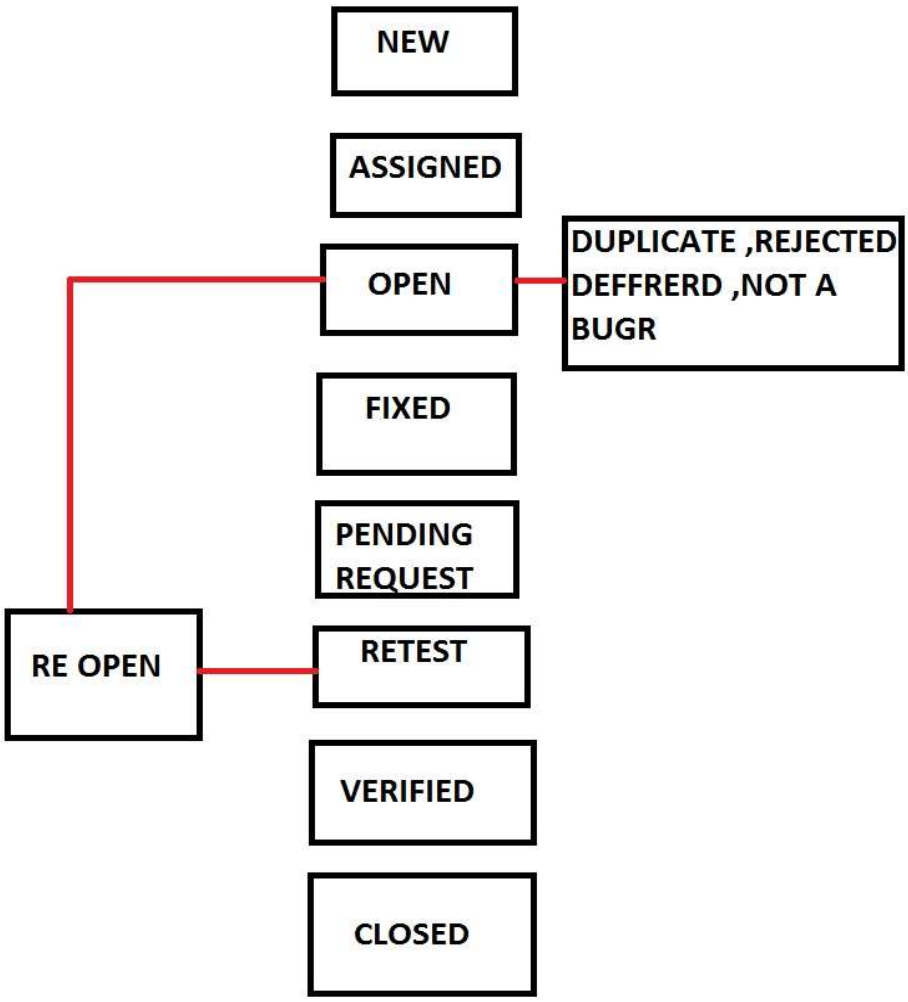
- It improves the quality of the product.
- It enhances the communication between the developing team and the testing team.
- It has the capability to adapt to multiple situations.

- What are the different Methodologies in Agile Development Model?

1. Kanban
2. Scrum
3. Extreme Programming (XP)
4. Feature-Driven Development
5. Crystal Methodology
6. Lean (Bonus)

- What is Bug Life Cycle?

Defect Life Cycle or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.


NEW
ASSIGNED
OPEN
DUPLICATE ,REJECTED DEFFRERD ,NOT A BUGR
FIXED
PENDING REQUEST
RE OPEN
RETEST
VERIFIED
CLOSED

- Difference between Priority and Severity ?

- Priority is the order in which the developer should resolve a defect whereas Severity is the degree of impact that a defect has on the operation of the product.
- Priority is categorized into three types: low, medium and high whereas Severity is categorized into five types: critical, major, moderate, minor and cosmetic.
- Priority is associated with scheduling while Severity is associated with functionality or standards.
- Priority indicates how soon the bug should be fixed whereas Severity indicates the seriousness of the defect on the product functionality.
- Priority of defects is decided in consultation with the manager/client while Severity levels of the defects are determined by the QA engineer.
- Priority value is subjective and can change over a period of time depending on the change in the project situation whereas Severity 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 while High Severity and low priority status indicates defect have to be fixed but not on immediate bases.
- Priority status is based on customer requirements whereas Severity status is based on the technical aspect of the product.

- Bug categories are...?

7 Common Types of Software Bugs Every Tester Should Know

1. Bug database defect
2. Critical functionality defect
3. Functionality defect
4. Security defect
5. User interface defect

- What is priority ?

One can define Priority as a parameter for deciding the order in which one can fix the defect. In this, the defect with a higher priority first needs to get fixed. Priority basically defines the order in which one would resolve any given defect.

- What is severity?

define Severity as the extent to which any given defect can affect/ impact a particular software. Severity is basically a parameter that denotes the impact of any defect and its

implication on a software's functionality. In other words, Severity defines the overall impact that any defect can have on a system.

- 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.