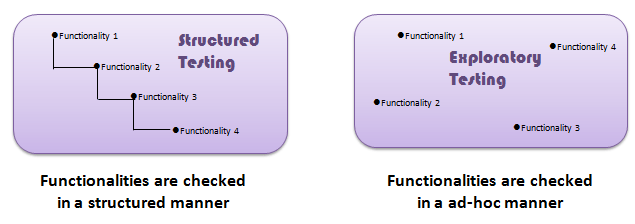
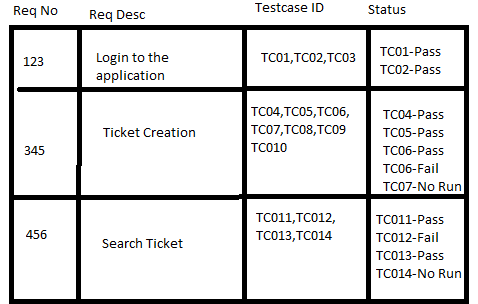
1. What is Exploratory Testing?

* **Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.
* Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.
* **Exploratory Testing Techniques**
* Is not random testing but it is ad-hoc testing with a purpose of find bugs
* Is structured and rigorous
* Is cognitively (thinking) structured as compared to the procedural structure of scripted testing. This structure comes from Charter, time boxing etc.
* Is highly teachable and manageable
* It is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently

1. What is traceability matrix?

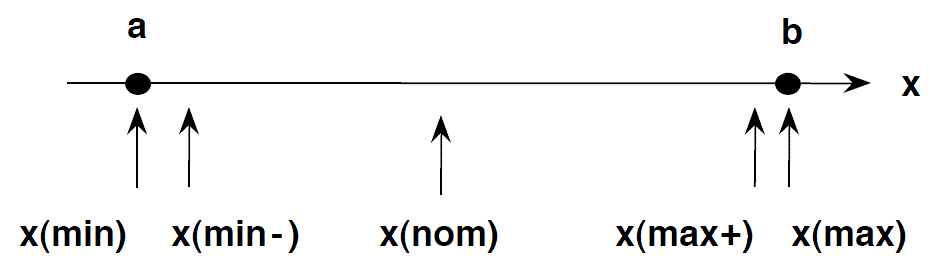
* A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.
* It is used to track the requirements and to check the current project requirements are met.
* **Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirements with test cases.
* It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle.
* The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.
* Parameters to include in requirements traceability matrix:
  + Requirement ID
  + Requirement Type and Description
  + Test Cases with Status

Above is a sample requirement traceability matrix.

1. What is Boundary value testing?

* BVT is based on testing the boundary values of valid and invalid partitions.
* The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundadaries are an area where testing is likely to yield defects.
* It checks for the input values near the boundary that have a higher chance of error .
* Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of the partition.
* Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.
* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.
* The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



* Limitation of Boundary Value Analysis:
  + It works well when the product is under test.
  + It cannot  consider the nature of the functional dependencies of variable.
  + Boundary value analysis is quite rudimentary.

4. What is Equivalence partitioning testing?

* Equivalence Partitioning is also called equivalence class partitioning.
* It is abbreviated as ECP.
* It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.
* An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.
* Equivalence Partitioning or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of [software testing](https://www.guru99.com/software-testing.html) like unit, integration, system, etc.
* In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of the small number of test cases.
* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

Example 1: Equivalence and Boundary Value

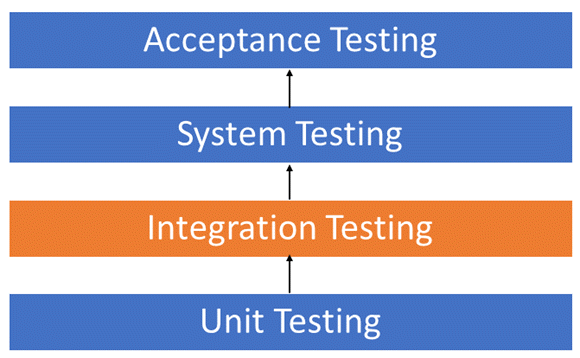
* Let’s consider the behavior of Order Pizza Text Box Below
* Pizza values 1 to 10 is considered valid. A success message is shown.
* While value 11 to 99 are considered invalid for order and an error message will appear, “Only 10 Pizza can be ordered”

Here is the test condition

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

5. What is Integration testing?

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



Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

Types of Integration Testing

Software Engineering defines the variety of strategies to execute Integration testing, viz.

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

Big Bang Testing

* 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.

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 link to be tested could be missed easily.
* Since the Integration testing can commence only after “all” the modules are designed, the 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.

Incremental Testing

* In the Incremental Testing approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application.
* Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

Incremental Approach, in turn, is carried out by two different Methods:

* Bottom Up
* Top Down

Advantages:

* Fault localization is easier.
* No time  is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

* Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
* An early prototype is not possible

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 organization and its stakeholder, should it occur.
* 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 may be categorised based on the guidelines set out below:
  + High
  + Medium
  + Low

**High**

* The impact of this risk would be very damaging and potentially non-tolerable. Ultimately, a risk of this scale could see the organisation make a loss should it occur. If a high risk is identified and cannot be solved, the software project may be too big of a risk to complete.

**Medium**

* Problems, challenges or glitches labeled as ‘medium risk’ may be tolerable but are certainly not desirable. These risks may see financial loss in the short-term, however, if a solution can be found, the positives of finishing the software project will outweigh the negative risks.

**Low**

* ‘Low’ risks can almost be classed as inconveniences or minor snags rather than actual threats to the project. Little to no financial loss would be seen in the event of one of these risks playing out.

7. What is Alpha testing?

* **Alpha Testing** is a type of software testing performed to identify bugs before releasing the software product to real users or public.
* It is a type of [acceptance testing.](https://www.guru99.com/user-acceptance-testing.html)
* The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

**Alpha Testing Process:**

1. Review the design specification and functional requirements.
2. Develop comprehensive test cases and test plans.
3. Execute test plan
4. Log defects
5. Retest once the issues have been fixed

**Phases of Alpha Testing:**

There are two phases in alpha testing:

1st Phase: The first phase of testing is done by in-house developers or software engineers. They either use hardware-aided debuggers or debugger software. The aim is to catch bugs quickly. Usually while alpha testing, a tester comes across to lots of bugs, crashes, missing features, and docs.

2nd Phase: The second phase of alpha testing is done by software quality assurance staff for additional testing in an environment. It includes a black box as well as white box testing.

**Advantages of Alpha Testing:**

* Better insight about the software’s reliability at its early stages.
* Free up your team for other projects.
* It reduces delivery time to market.
* Early feedback helps to improve software quality.

**Disadvantages of Alpha Testing:**

* It will need a longer time for test plan execution if the project is large.
* Sometimes, the defects in the products can be unknown during this alpha testing.
* It is difficult to test the entire product since it is still under development.
* For smaller projects, time spent on alpha testing is not worthy enough.
* It does not carry out reliability and security testing.
* This test will only cover the business requirements mentioned by the client. The project team will not go through the deep testing of each and every module.
* It requires a separate lab environment for testing.

8. What is Beta testing?

* Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of User Acceptance Testing.
* A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality.
* Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation.
* It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

**Characteristics of Beta Testing:**

1. Beta Testing is performed by clients or users who are not employees of the company.
2. Reliability, security, and robustness are checked during beta testing.
3. Beta Testing commonly uses black-box testing.
4. Beta testing is carried out in the user’s location.
5. Beta testing doesn’t require a lab or testing environment.

**Types of Beta Testing:**

There are different types of beta testing:

1. Traditional Beta testing: Product is distributed to the target market and related data is gathered in all aspects. This data can be used for Product improvement.
2. Public Beta Testing: Product is released publicly to the world through online channels and data can be collected from anyone. Based on feedback, product improvements can be done. For example, Microsoft conducted the largest of all Beta Tests for its operating system Windows 8 before officially releasing it.
3. Technical Beta Testing: Product is released to a group of employees of an organization and collects feedback/data from the employees of the organization.
4. Focused Beta Testing: Software product is released to the market for collecting feedback on specific features of the program. For example, important functionality of the software.
5. Post-release Beta Testing: Software product is released to the market and data is collected to make improvements for the future release of the product.

**Criteria for Beta Testing:**

* Sign off a document on Alpha testing.
* The Beta version of the software should be ready
* Environment ready to release the software application to the public
* Tool to capture real-time faults

**Tools used for Beta Testing:**

* TestFairy
* CenterCode
* TryMyUI
* UserTesting
* TestRail
* Usersnap
* Zephyr
* TestFlight

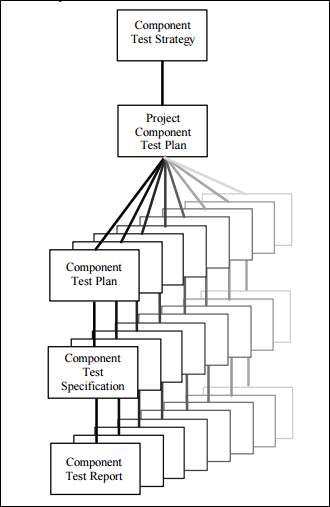
**Advantages of Beta Testing:**

* It reduces product failure risk via customer validation.
* Beta Testing allows a company to test post-launch infrastructure.
* It helps in improving product quality via customer feedback.
* Cost-effective compared to similar data gathering methods.
* It creates goodwill with customers and increases customer satisfaction.

**Disadvantages of Beta Testing:**

* Sometimes, it is complex to follow the errors or bugs because the testing environment varies from user to user.
* There is a chance of having duplication of errors or bugs.
* The development team and the testing team are not having control over this real-time test environment.
* This testing is a time consuming process since it involves real time users or clients and hence delay in the overall feedback about the entire product.
* The users who are testing the product should have enough knowledge about the working of the entire application or product. Otherwise, the testing will not be implemented in an efficient manner.

9. 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.

**Component Testing**

**Component Testing Techniques**

* Based on depth of testing levels, Component testing can be categorized as

1. CTIS – Component Testing In Small
2. CTIL – Component Testing In Large

* CTIS – Component Testing in Small
* Component testing may be done with or without isolation of rest of other components in the software or application under test.
* If it’s performed with the isolation of other component, then it’s referred as Component Testing in Small.
* Example 1: Consider a website which has 5 different web pages then testing each webpage separately & with the isolation of other components is referred as Component testing in Small.
* Example 2: Consider the home page of the guru99.com website which has many components like
* Home, Testing, SAP, Web, Must Learn!, Big Data, Live Projects, Blog and etc.
* Similarly, any software is made of many components and also, every component will have its own subcomponents.
* Testing each modules mentioned in example 2 separately without considering integration with other components is referred as Component Testing in Small.

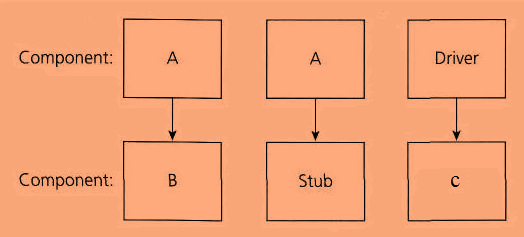
**CTIL – Component Testing in Large**

Component testing done without isolation of other components in the software or application under test is referred as Component Testing Large.

Let’s take an example to understand it in a better way.

Suppose there is an application consisting of three components say **Component A**, **Component B,** and **Component C**.

The developer has developed the component B and wants it tested. But in order to **completely** test the component B, few of its functionalities are dependent on component A and few on component C.



Functionality Flow: **A** -> B -> **C** which means there is a dependency to B from both A & C, as per the diagram stub is the **called function,** and the driver is the **calling function**.

But the component A and component C has not been developed yet. In that case, to test the component B completely, we can replace the component A and component C by stub and drivers as required. So basically, component A & C are replaced by stub & driver’s which acts as a dummy object till they are actually developed.

* **Stub:** A stub is called from the software component to be tested as shown in the diagram below ‘Stub’ is called by Component A.
* **Driver:** A driver calls the component to be tested as shown in the diagram below ‘Component B’ is called by Driver.

10. 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.

What do you test in Functional Testing?

* **Mainline functions**:  Testing the main functions of an application
* **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
* **Accessibility**:  Checks the accessibility of the system for the user
* **Error Conditions**: Usage of testing techniques to check for error conditions.

It checks whether suitable error messages are displayed.

How to do Functional Testing:

* Understand the Functional Requirements
* Identify test input or test data based on requirements
* Compute the expected outcomes with selected test input values
* Execute test cases
* Compare actual and computed expected results

11. 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.

Objectives of Non-functional testing

* Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
* Helps to reduce production risk and cost associated with non-functional aspects of the product.
* Optimize the way product is installed, setup, executes, managed and monitored.
* Collect and produce measurements, and metrics for internal research and development.
* Improve and enhance knowledge of the product behavior and technologies in use.

Characteristics of Non-functional testing

* Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
* Exact numbers are unlikely to be known at the start of the requirement process
* Important to prioritize the requirements
* Ensure that quality attributes are identified correctly in Software Engineering.

Non-functional testing Parameters:

**1) Security:**

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via [Security Testing](https://www.guru99.com/what-is-security-testing.html).

**2) Reliability:**

The extent to which any software system continuously performs the specified functions without failure. This is tested by [Reliability Testing](https://www.guru99.com/reliability-testing.html)

**3) Survivability:**

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by [Recovery Testing](https://www.guru99.com/recovery-testing.html)

**4) Availability:**

The parameter determines the degree to which user can depend on the system during its operation. This is checked by [Stability Testing.](https://www.guru99.com/stability-testing.html)

**5) Usability:**

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html)

**6) Scalability:**

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by [Scalability Testing](https://www.guru99.com/scalability-testing.html)

**7) Interoperability:**

This non-functional parameter checks a software system interfaces with other software systems. This is checked by [Interoperability Testing](https://www.guru99.com/interoperability-testing.html)

**8) Efficiency:**

The extent to which any software system can handles capacity, quantity and response time.

**9) Flexibility:**

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

**10) Portability:**

The flexibility of software to transfer from its current hardware or software environment.

**11) Reusability:**

It refers to a portion of the software system that can be converted for use in another application.

12. What is GUI Testing?

* **GUI Testing** is a software testing type that checks the Graphical User Interface of the Software.
* The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
* A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

**GUI Testing Tools** :

1. Selenium
2. QTP
3. Cucumber
4. SilkTest
5. TestComplete
6. Squish GUI Tester

13. What is adhoc testing?

* **Adhoc 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.
* Adhoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.
* Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
* Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called **Error Guessing.**
* Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.
* This testing requires no documentation/ planning /process to be followed.
* Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases.
* This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

**Adhoc testing has –**

* No Documentation.
* No Test cases.
* No Test Design.

**Types of Adhoc testing:**

* Buddy Testing
* Pair Testing
* Monkey Testing

14. 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.

This testing usually identifies –

* The maximum operating capacity of an application
* Determine whether the current infrastructure is sufficient to run the application
* Sustainability of application with respect to peak user load
* Number of concurrent users that an application can support, and scalability to allow more users to access it.

Why Load Testing?

* Load testing gives confidence in the system & its reliability and performance.
* Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.
* Load testing gives excellent protection against poor performance and accommodates complementary strategies for performance management and monitoring of a production environment.

Goals of Load Testing:

Loading testing identifies the following problems before moving the application to market or Production:

* Response time for each transaction
* Performance of System components under various loads
* Performance of Database components under different loads
* Network delay between the client and the server
* Software design issues
* Server configuration issues like a Web server, application server, database server etc.
* Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

Prerequisites of load testing:

The chief metric for load testing is response time. Before you begin load testing, you must determine –

* Whether the response time is already measured and compared – Quantitative
* Whether the response time is applicable to the business process – Relevant
* Whether the response time is justifiable – Realistic
* Whether the response time is achievable – Achievable
* Whether the response time is measurable using a tool or stopwatch – Measurable

15. 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.

Need for Stress Testing:

Consider the following real-time examples where we can discover the usage of Stress Testing-

* During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
* When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.

Stress testing is also extremely valuable for the following reasons:

* To check whether the system works under abnormal conditions.
* Displaying appropriate error message when the system is under stress.
* System failure under extreme conditions could result in enormous revenue loss
* It is better to be prepared for extreme conditions by executing Stress Testing.

Goals of Stress Testing:

* The goal of stress testing is to analyze the behavior of the system after a failure. For stress testing to be successful, a system should display an appropriate error message while it is under extreme conditions.
* To conduct Stress Testing, sometimes, massive data sets may be used which may get lost during Stress Testing. Testers should not lose this security-related data while doing stress testing.
* The main purpose of stress testing is to make sure that the system recovers after failure which is called as **recoverability**.

Types of Stress Testing:

* **Distributed Stress Testing:**
  + The role of stress server is to distribute a set of stress tests to all stress clients and track on the status of the client. After the client contacts the server, the server adds the name of the client and starts sending data for testing.
* **Application Stress Testing:**
  + This testing concentrate on finding defects related to data locking and blocking, network issues and performance bottlenecks in an application.
* **Transactional Stress Testing:**
  + It does stress testing on one or more transactions between two or more applications. It is used for fine-tuning & optimizing the system.
* **Systemic Stress Testing:**
  + This is integrated stress testing which can be tested across multiple systems running on the same server. It is used to find defects where one application data blocks another application.
* **Exploratory Stress Testing:**
  + This is one of the types of stress testing which is used to test the system with unusual parameters or conditions that are unlikely to occur in a real scenario. It is used to find defects in unexpected scenarios like

1. A large number of users logged at the same time
2. If a virus scanner started in all machines simultaneously
3. If Database has gone offline when it is accessed from a website,
4. When a large volume of data is inserted to the database simultaneously

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

* White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.
* It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

White Box Testing Techniques

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage
* Multiple Condition Coverage
* Finite State Machine Coverage
* Path Coverage
* Control flow testing
* Data flow testing

Types of White Box Testing:

* **Unit Testing:** It is often the first type of testing done on an application. [Unit Testing](https://www.guru99.com/unit-testing-guide.html) is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
* **Testing for Memory Leaks**: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Apart from the above, a few testing types are part of both black box and white box testing. They are listed below:

* **White Box** [**Penetration Testing**](https://www.guru99.com/learn-penetration-testing.html)**:** In this testing, the tester/developer has full information of the application’s source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats.
* **White Box Mutation Testing**: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

Advantages of White Box Testing

* Code optimization by finding hidden errors.
* White box tests cases can be easily automated.
* Testing is more thorough as all code paths are usually covered.
* Testing can start early in SDLC even if GUI is not available.

Disadvantages of WhiteBox Testing

* White box testing can be quite complex and expensive.
* Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed and can lead to production errors.
* White box testing requires professional resources with a detailed understanding of programming and implementation.
* White-box testing is time-consuming, bigger programming applications take the time to test fully.

17. What is Black box testing? What are the different between black box testing technique.

* **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.



Black Box Testing Techniques

Following are the prominent [Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html) amongst the many used in Black box Testing

* **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.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones –

* **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
* **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
* **Regression testing** – [Regression Testing](https://www.guru99.com/regression-testing.html) is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Black Box Testing and Software Development Life Cycle (SDLC)

Black box testing has its own life cycle called Software Testing Life Cycle ([STLC](https://www.guru99.com/software-testing-life-cycle.html)) and it is relative to every stage of Software Development Life Cycle of Software Engineering.

* **Requirement** – This is the initial stage of SDLC and in this stage, a requirement is gathered. Software testers also take part in this stage.
* **Test Planning & Analysis** – [Testing Types](https://www.guru99.com/types-of-software-testing.html) applicable to the project are determined. A [Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html) is created which determines possible project risks and their mitigation.
* **Design** – In this stage Test cases/scripts are created on the basis of software requirement documents
* **Test Execution**– In this stage Test Cases prepared are executed. Bugs if any are fixed and re-tested.

18. Mention what are the categories of defects

* **Error of Commission:**
  + Commission means instruction or some kind of command given. Now the error in commission means the error in made in command or instruction.
  + For example, suppose I wrote a loop which I was trying to run 10 times but I command it to run more than 10 times by mistake this is the error of commission.
* **Errors of Omissions:**
  + As name is already describing error of omission is some thing which happens accidentally. Omission word means something left out or executed.
  + Practical most common example of this error is suppose we make a function in programming open its bracket but forget to close at the end.
* **Error of Clarity:**
  + The most common error in the natural languages.
  + This error happens due to miss understanding between the developer and client. It travels most of the time from the requirements to the software.
* **Error of Speed or Capacity:**
  + The name of the error is itself enough i think to tell about it this error.
  + Your software is working fine but not working in the required time this is the error of speed.
  + When it comes to capacity it can be relevant to memory. For example, a small integer is declared where the long integer was required.

19. What is bigbang testing?

* Big bang integration testingis a testing approach where all components or modules are integrated and tested as a single unit.
* This is done after all modules have been completed and before any system-level testing is performed.
* This is in contrast to incremental integration testing, in which components are tested one at a time or in small groups.
* This approach is typically used when there is a tight deadline for delivering the software product, and all development teams are working in parallel on their respective components.
* For example, consider a simple system with three modules A, B, and C. Module A has been tested and found to be working correctly. The same is true for modules B and C. To test the system as a whole, all three modules are integrated and tested together.

Features of Big Bang Integration Testing

* **Simulation of the complete system:** Big bang integration testing involves simulating the complete system. This means that all components and modules are integrated and tested at the same time.
* **Testing all components together:** Since all components are integrated and tested at the same time, this means that all components are tested together. This is beneficial as it allows for the testing of interactions between components.
* **No component is left untested:** Since all components are integrated and tested together, this means that no component is left untested. This is beneficial as it ensures that all aspects of the system are tested.
* **Early detection of errors:** Big bang integration testing can detect errors early on in the development process. This is beneficial as it allows for the correction of errors before the system is deployed.
* **Allows for testing of complex interactions:** Big bang integration testing allows for testing of complex interactions between components. This is beneficial as it allows for the identification of errors that may not be detected by other testing methods.
* **Simulate the behavior of lower-level components:** Big bang integration testing uses stubs and drivers to simulate the behavior of lower-level components.
* **Top-down approach:** It is also known as a top-down approach because testing starts from the highest level component and moves down the component hierarchy.
* **The basic form of integration testing:** It is the most basic form of integration testing, where all components are integrated and tested together.
* **Risky:** It can be very risky, as all components need to be working correctly for the system to work correctly. This approach is not recommended for large or complex projects.
* **End of the development cycle:** This type of testing is usually done at the end of the development cycle when all the modules are ready to be integrated. It can be used to test the functionality of the system as a whole.
* **Manual:** This type of testing is usually done manually, as it can be difficult to automate all the modules at once.
* **Time-consuming and expensive:** It can be time-consuming and expensive, as all the modules need to be tested together.

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.
* Software testing teams will use exit criteria to determine if a test plan or project can exit to the next stage or be considered complete.
* This isn't something that should be left up to the subjective and/or ad hoc decisions of a test admin or SQA engineer, as it can directly impact the success of the next stage or project as a whole.

Creating exit criteria helps:

* Align your teams on a common definition of test completion
* Ensure your product meets completion standards before entering the next stage, which avoids costly project delays
* Create clear parameters for test engineers to evaluate software.

21. When should “regression testing” be performed?

Typically, regression testing is performed under these circumstances:

* A new requirement is added to an existing feature
* A new feature or functionality is added
* The codebase is fixed to solve defects
* The source code is optimized to improve performance
* Patch fixes are added
* Changes in configuration

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

7 Kye principles are listed below:

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**

* Yes! Exhaustive testing is not possible. 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.

**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 defects.

**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 machine.

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

|  |  |  |
| --- | --- | --- |
| **Quality Assurance** | **Quality Control** | **Testing** |
| 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. |

24. Difference between smoke and sanity?

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of [Regression Testing](https://www.guru99.com/regression-testing.html) |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

25. Difference between verification and validation

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| The verifying process includes checking documents, design, code, and program | It is a dynamic mechanism of testing and validating the actual product |
| It does ***not*** involve executing the code | It always involves executing the code |
| Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc. | It uses methods like Black Box Testing, [White Box Testing](https://www.guru99.com/white-box-testing.html), and non-functional testing |
| Whether the software conforms to specification is checked | It checks whether the software meets the requirements and expectations of a customer |
| It finds bugs early in the development cycle | It can find bugs that the verification process can not catch |
| Target is application and software architecture, specification, complete design, high level, and database design etc. | Target is an actual product |
| QA team does verification and make sure that the software is as per the requirement in the SRS document. | With the involvement of testing team validation is executed on software code. |
| It comes before validation | It comes after verification |

26. 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”***.

Types of Performance Testing:

* **Load testing –** checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing –** tests the software’s reaction to sudden large spikes in the load generated by users.
* **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
* **Scalability testing** – The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

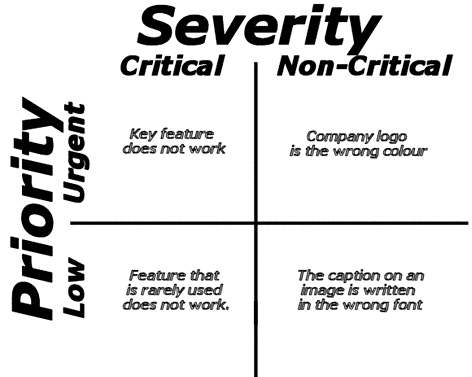
The focus of Performance Testing is checking a software program’s

* **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

27. What is error, defect, bug and failure?

* **Bug:** A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which results in a bug. It is now that the Automation/ Manual Test Engineers describe this situation as a bug.  
  A bug once detected can be reproduced with the help of standard bug reporting templates.
* **Defect:** A defect refers to the situation when the application is not working as per the requirement and the actual and expected result of the application or software are not in sync with each other.
* **Error:** Error is a situation that happens when the Development team or the developer fails to understand a requirement definition and hence that misunderstanding gets translated to buggy code. This situation is referred to as an Error and is mainly a term coined by the developers.
* **Failure:** Failure is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive. Generally, such situations happen very rarely because before releasing a product all possible scenarios and test cases for the code are simulated.  Failure is detected by end-users once they face a particular issue in the software.

28. 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 |

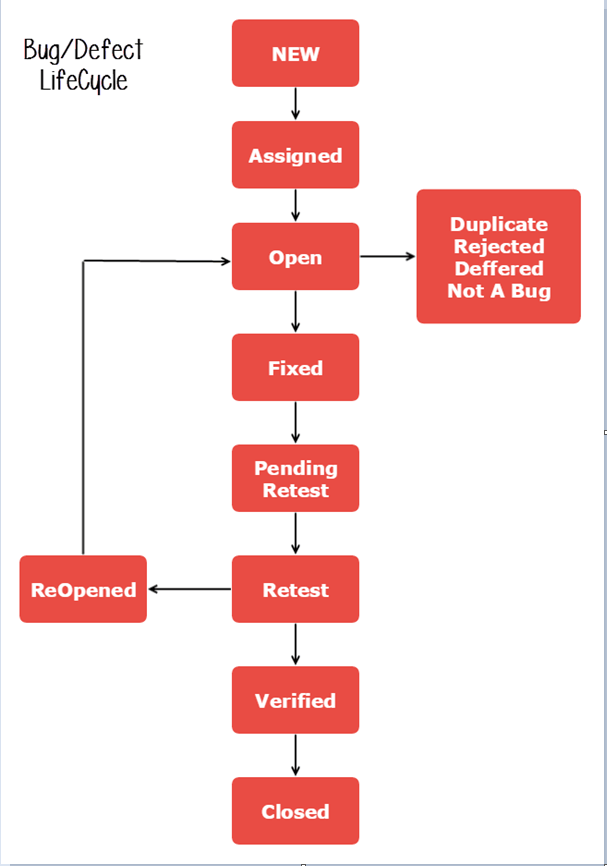
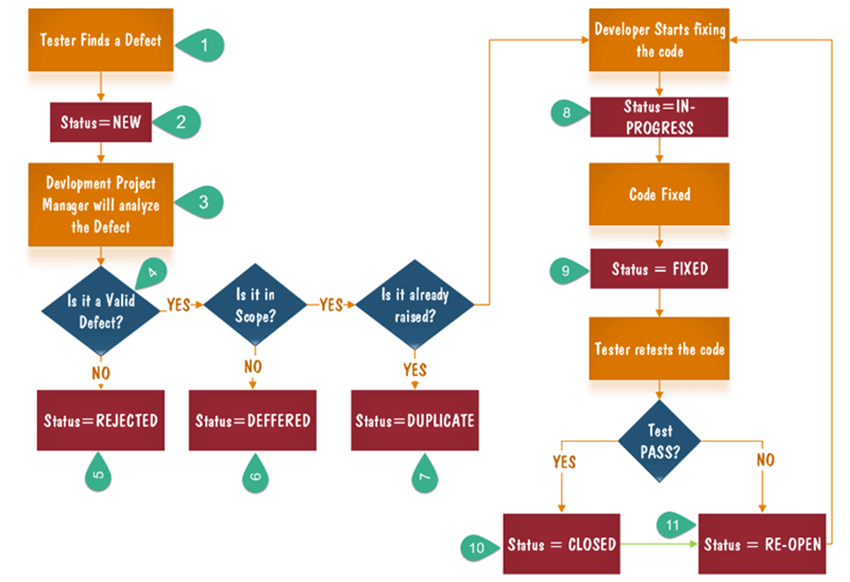
29. What is bug Life cycle?

* **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.
* **Defect Status** or Bug Status in defect life cycle is the present state from which the defect or a bug is currently undergoing.
* The goal of defect status is to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.

**Defect States Workflow**

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

1. Tester finds the defect
2. Status assigned to defect- New
3. A defect is forwarded to Project Manager for analyze
4. Project Manager decides whether a defect is valid
5. Here the defect is not valid- a status is given “Rejected.”
6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred** status.
7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
9. Once the code is fixed. A defect is assigned a status **fixed**
10. Next, the tester will re-test the code. In case, the [Test Case](https://www.guru99.com/test-case.html) passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

30. 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. |
| 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 |

31. 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. |

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

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Test Scenario** | **Test Script** |
| A test case contains clearly defined test steps for testing a feature of an application. | A test scenario contains a high level documentation, describing an end to end functionality to be tested. | A set of instructions that are intended to test if the functionality of the software is working properly |
| Test cases focus on “what to test” and “how to test”. | Test scenarios just focus on “what to test”. | To perform an automation test based on a specific requirement on the script |
| These have clearly defined step, pre-requisites, expected results etc. Hence, there is no ambiguity. | Test scenarios are generally one-liner. Hence, there is always possibility of ambiguity during testing. | Programming language |
| Test cases can be derived from test scenarios. They have many to one relationship with the test scenarios. | These are derived from use cases. | Automation testing environment |
| More resources are required for documentation and execution of test cases. | Relatively less time and resources are required for creating and testing using scenarios. | * Helps meet user and business requirements * Validates the functionality of the software * Allows performing fast and efficient automated tsting * Can be reusable in the future |

33. What is test plan? 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.

Definition:  **“Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”**

* A test plan is a document that outlines the objectives, scope, approach, and schedule of testing activities for a software project or system. It is a comprehensive roadmap that defines how the testing process will be carried out, including what will be tested, how it will be tested, who will do the testing, when it will be done, and how the results will be reported.

A typical test plan includes the following information:

1. Introduction: A brief overview of the testing process and the purpose of the test plan.
2. Test Objectives: The specific goals and objectives of the testing process, including what aspects of the software or system will be tested.
3. Test Strategy: The overall approach to testing, including the types of testing that will be performed, the tools and techniques that will be used, and the testing environment.
4. Test Scope: The scope of testing, including what features or functionality will be tested, and what will not be tested.
5. Test Schedule: A timeline or schedule for testing activities, including key milestones, deadlines, and dependencies.
6. Test Deliverables: A list of the artifacts and documentation that will be produced as part of the testing process.
7. Test Resources: The people, tools, and other resources required for testing, including testing personnel, hardware, software, and other resources.
8. Test Risks: An assessment of the risks associated with the testing process, including potential issues or challenges that may impact the testing process.
9. Test Reporting: A description of the reporting process, including how test results will be documented, tracked, and reported.

34. What is priority?

* Priority is a measure of the relative importance or urgency of a task or issue.
* In software development and testing, priority is used to determine the order in which issues or bugs should be addressed.
* Priority is usually assigned based on factors such as the impact of the issue on the system or users, the severity of the issue, and the potential risks associated with leaving the issue unresolved.
* The higher the priority of an issue, the more urgent it is to address it.
* In most cases, priority is expressed using a numerical scale or a set of labels, such as "high", "medium", or "low".
* For example, a critical issue that is causing the system to crash might be assigned a priority of "1" or "high", while a minor issue that has little impact on the system might be assigned a priority of "3" or "low".
* Assigning priority helps ensure that critical issues are addressed first, minimizing the impact on users and reducing the risk of further problems.
* It also helps developers and testers prioritize their work and allocate resources effectively.
* 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 Types

Types of Priority of bug/defect can be categorized into three parts :

* **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.

35. What is severity?

* Severity is a measure of the impact or seriousness of a defect or issue in a software system.
* It is used to indicate the level of impact that the defect has on the system's functionality, performance, or usability.
* In software development and testing, severity is typically assigned by testers or quality assurance personnel based on the potential impact of the issue.
* The severity level can range from minor issues that have little or no impact on the system to critical issues that can cause the system to fail or put user data at risk.

Common severity levels used in software testing include:

1. Critical: Issues that cause the system to crash or cause severe data loss, which can result in a complete breakdown of the system.
2. High: Issues that significantly impact the functionality of the system and can cause major problems for users.
3. Medium: Issues that have a moderate impact on the system's functionality and can affect the user experience.
4. Low: Minor issues that have little or no impact on the system's functionality or the user experience.

* By assigning severity levels to defects or issues, testers and developers can prioritize their work and allocate resources more effectively.
* High and critical severity issues are typically given the highest priority and are addressed as quickly as possible to minimize the impact on users and the system.
* Severity status is based on the technical aspect of the product.
* High severity and low priority status indicates defect have to be fixed but not on immediate bases

Types of Severity

In Software Testing, Types of Severity of bug/defect can be categorized into the following parts:

* **Critical**: This defect indicates complete shut-down of the process, nothing can proceed further
* **Major**: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional
* **Medium**: It causes some undesirable behavior, but the system is still functional
* **Low**: It won’t cause any major break-down of the system

36. Bug categories are ..

1. Functional bugs: These bugs relate to issues with the functional requirements of the software. They occur when the software does not perform the intended function or does not meet the specified requirements.
2. Performance bugs: These bugs relate to issues with the performance of the software. They occur when the software does not meet performance requirements, such as speed, memory usage, or scalability.
3. Usability bugs: These bugs relate to issues with the usability of the software. They occur when the software is difficult to use, confusing, or does not meet the user's expectations.
4. Compatibility bugs: These bugs relate to issues with the compatibility of the software with other systems, such as hardware, software, or network configurations.
5. Security bugs: These bugs relate to issues with the security of the software. They occur when the software is vulnerable to security threats, such as hacking, data breaches, or unauthorized access.
6. Configuration bugs: These bugs relate to issues with the configuration of the software. They occur when the software is not configured correctly, resulting in errors or unexpected behavior.
7. Documentation bugs: These bugs relate to issues with the documentation of the software. They occur when the documentation is incomplete, inaccurate, or does not match the actual behavior of the software.

37. Advantage of bugzilla

1. Centralized bug tracking: Bugzilla provides a centralized repository for tracking bugs and issues, making it easier for developers and testers to track the status of issues and collaborate on their resolution.
2. Customizable workflows: Bugzilla offers a flexible workflow system that can be customized to match the specific needs of different projects or teams. This allows teams to define their own workflows and processes for bug tracking and resolution.
3. Integration with other tools: Bugzilla can be integrated with other software development tools, such as version control systems and project management tools, making it easier for teams to manage their entire development process from a single system.
4. Reporting and analysis: Bugzilla provides a range of reporting and analysis tools, allowing teams to generate custom reports and charts to analyze bug data and track progress over time.
5. User-friendly interface: Bugzilla's user interface is intuitive and easy to use, making it easy for team members to submit, track, and manage bugs and issues.
6. Open-source and free: Bugzilla is an open-source software tool that is freely available for use by anyone. This makes it an accessible and cost-effective solution for software development and testing teams of all sizes.

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

* Agile development is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and rapid delivery of working software.
* There are several methodologies within the Agile development model, including:

1. Scrum: Scrum is a popular Agile methodology that emphasizes collaboration, accountability, and iterative development. It involves dividing development work into short sprints, typically 2-4 weeks in duration, and using daily stand-up meetings to keep team members aligned and focused.
2. Kanban: Kanban is an Agile methodology that emphasizes visualization, flow, and continuous improvement. It involves using a Kanban board to visualize work items and track progress through the development pipeline.
3. Extreme Programming (XP): XP is an Agile methodology that emphasizes rapid feedback, continuous testing, and continuous integration. It involves practices such as pair programming, test-driven development, and frequent releases.
4. Lean Agile: Lean Agile is an Agile methodology that combines principles from Agile development and lean manufacturing. It involves minimizing waste, optimizing flow, and maximizing value to the customer.
5. Crystal: Crystal is an Agile methodology that emphasizes communication, simplicity, and frequent delivery. It involves using a set of lightweight processes and practices that are tailored to the specific needs of the project and team.
6. Feature Driven Development (FDD): FDD is an Agile methodology that emphasizes feature delivery, teamwork, and continuous inspection. It involves breaking development work into small, feature-focused chunks and using a structured process to guide development.

39. Difference between Authorization and authentication

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| In the [authentication](https://www.geeksforgeeks.org/authentication-in-computer-network/) process, the identity of users are checked for providing the access to the system. | While in [authorization](https://www.geeksforgeeks.org/what-is-aaa-authentication-authorization-and-accounting/) 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) * [Single sign-on (SSO)](https://www.geeksforgeeks.org/introduction-of-single-sign-on-sso/) * Social authentication | Popular  Authorization Techniques-   * Role-Based Access Controls (RBAC) * [JSON web token (JWT) Authorization](https://www.geeksforgeeks.org/json-web-token-jwt/) * 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. |

* What are the common problem faced in web testing?
  + Integration
  + Interoperability
  + Security
  + Performance
  + Usability
  + Quality Testing, Exceptional Services
  + Cross Browser Compatibility
  + Responsiveness
  + Cross-Device Compatibility
  + Accessibility Testing
  + Project Deadline