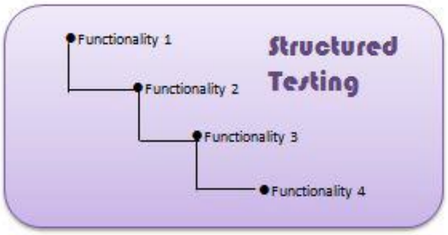
Software Testing Assignment

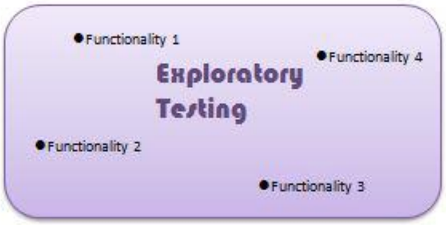
Module – 2(Manual Testing)

1) What is Exploratory Testing?

- Exploratory software testing is done on the fly and relies on the tester to think beyond the limits of the scripted tests. Exploratory testers design a test, execute it immediately, observe the results, and use results to design the next test.
- Exploratory testing is **an approach to software testing that is often described as simultaneous learning, test design, and execution**.
- Test design, execution and logging happen simultaneously
- Testing is often not recorded
- Makes use of experience, heuristics and test patterns
- Testing is based on a test charter that may include
- Scope of the testing (in and out)
- The focus of exploratory testing is more on testing as a “thinking” activity.
- A brief description of how tests will be performed
- Expected problems
- Is carried out in time boxed intervals
- **Though the current trend in testing is to push for automation**, exploratory testing is a new way of thinking. **Automation has its limits**
- Is not random testing but it is Adhoc testing with purpose of find bugs
- Is not a technique but it is an approach. What actions you perform next is governed by what you are doing currently



Functionalities are checked in a structured manner



Functionalities are checked in a ad-hoc manner

Types of Exploratory Testing

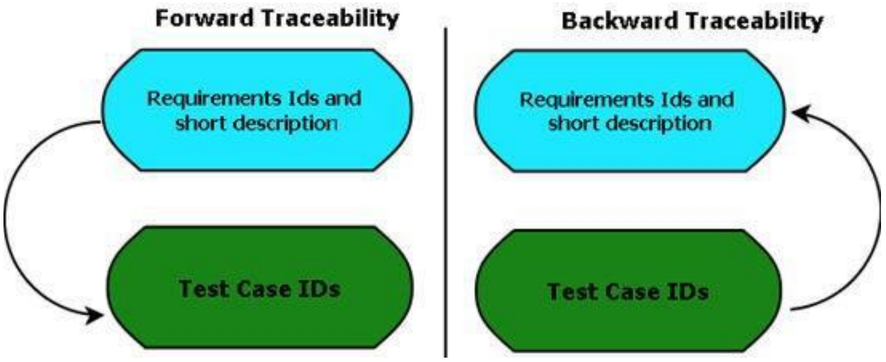
- Freestyle Exploratory Testing- Think of this as exploratory testing with an ad-hoc approach. In this format, there are no rules, structure, or organization. ...
- Scenario-based Exploratory Testing- This form of exploratory testing is based on real user scenarios. ...
- Strategy-based Exploratory Testing.

2) What is traceability matrix?

- Traceability matrix is **a table type document that is used in the development of software application to trace requirements**
- A traceability matrix is **a document that details the technical requirements for a given test scenario and its current state**.
- It helps the testing team understand the level of testing that is done for a given product. The traceability process itself is used to review the test cases that were defined for any requirement.

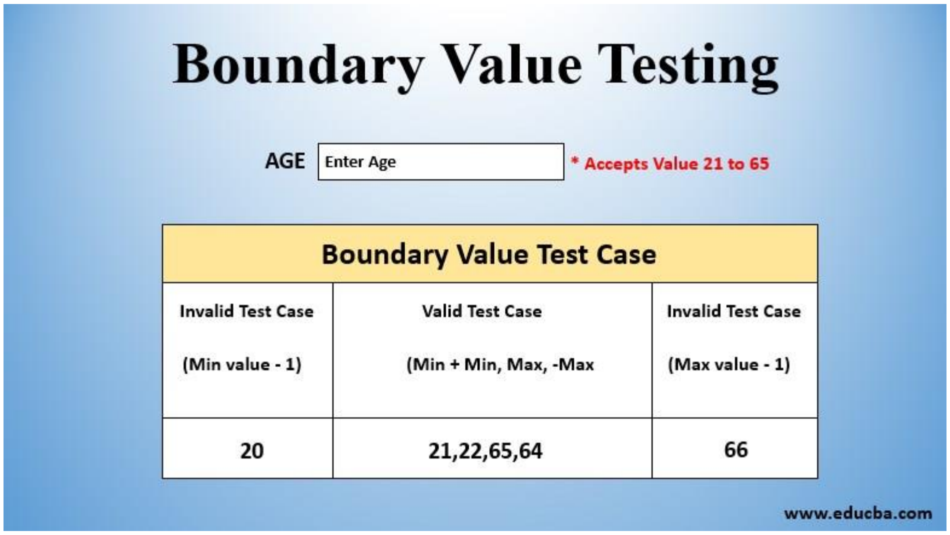
Types of requirements traceability

- Forward traceability- Mapping of Requirements to Test cases
- Backward traceability- Mapping of Test Cases to Requirements
- Bidirectional traceability - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.


Forward Traceability
Requirements Ids and short description
Test Case IDs
Backward Traceability
Requirements Ids and short description
Test Case IDs

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.
- Boundary value analysis generates test cases that highlight errors better than equivalence partitioning.
- The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes.
- At those points when input values change from valid to invalid errors are most likely to occur.



Boundary Value Testing

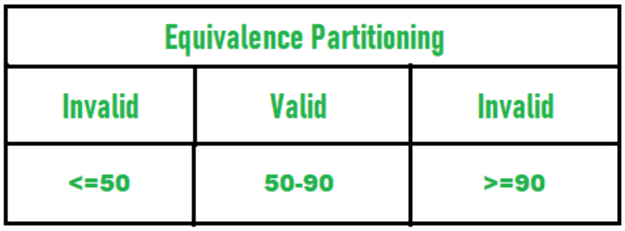
AGE Enter Age * Accepts Value 21 to 65

Boundary Value Test Case		
Invalid Test Case (Min value - 1)	Valid Test Case (Min + Min, Max, -Max	Invalid Test Case (Max value - 1)
20	21,22,65,64	66

www.educba.com

4) What is Equivalence partitioning testing?

- Equivalence partitioning is used **to reduce the number of test cases that are required to effectively test the handling of inputs, outputs, internal values, and time-related values**.
- Aim is to treat groups of inputs as equivalent and to select one representative input to test them all
- **Example-1**
  Consider percentage field that will accept percentage only between 50 to 90 %, more and even less than not be accepted, and application will redirect user to an error page.
  If percentage entered by user is less than 50 %or more than 90 %, that equivalence partitioning method will show an invalid percentage



Equivalence Partitioning		
Invalid	Valid	Invalid
<=50	50-90	>=90

5) What is Integration testing?

- Integration testing -- also known as integration and testing (I&T) -- is **a type of software testing in which the different units, modules or components of a software application are tested as a combined entity**. However, these modules may be coded by different programmers.
- Integration Testing - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems
- Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
- The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.
- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- Integration testing is done by a specific integration tester or test team.
- Components may be code modules, operating systems, hardware and even complete systems
- **There are 2 levels of Integration Testing**
- Component Integration Testing
- System Integration Testing
- **There is two types methods of Integration Testing:**
- Bing Bang Integration Testing
- Incremental Integration Testing

  Top down Approach

  Bottom up Approach
- **When is Integration Testing performed?**

  Integration Testing is performed after Unit Testing and before System Testing.
- **Who performs Integration Testing?**

  Either Developers themselves or independent Testers perform Integration Testing.

6) What determines the level of risk?

- Risk – **'A factor that could result in future negative consequences; usually expressed as impact and likelihood'**
- In Software Testing, risk analysis is the process of identifying the risks in applications or software that you built and prioritizing them to test. After that, **the process of assigning the level of risk is done**. The categorization of the risks takes place, hence, the impact of the risk is calculated
- When testing does find defects, the Quality of the software system increases when those defects are fixed
- The Quality of systems can be improved through Lessons learned from previous projects
- Analysis of root causes of defects found in other projects can lead to Process Improvement

- Process Improvement can prevent those defects reoccurring
- Which in turn, can improve the Quality of future systems
- Testing should be integrated as one of the Quality assurance activities

7) What is Alpha testing?

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

8) What is beta testing?

- **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.
- It is only a kind of Black Box Testing.
- Beta testing can be considered “**pre-release**” testing.
- **Pilot Testing** is testing to product on real world as well as collect data on the use of product in the classroom.

9) What is component testing?

- Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
- Component Testing – The testing of individual software components.
- Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
- A unit is the smallest testable part of an application like functions/procedures, classes, interfaces.
- Unit testing is the first level of testing and is performed prior to Integration Testing.

- Sometimes known as Unit Testing, Module Testing or Program Testing
- Component can be tested in isolation – stubs/drivers may be employed
- Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing.
- Functional and Non-Functional testing
- Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.
- Unit testing is performed by using the White Box Testing method.

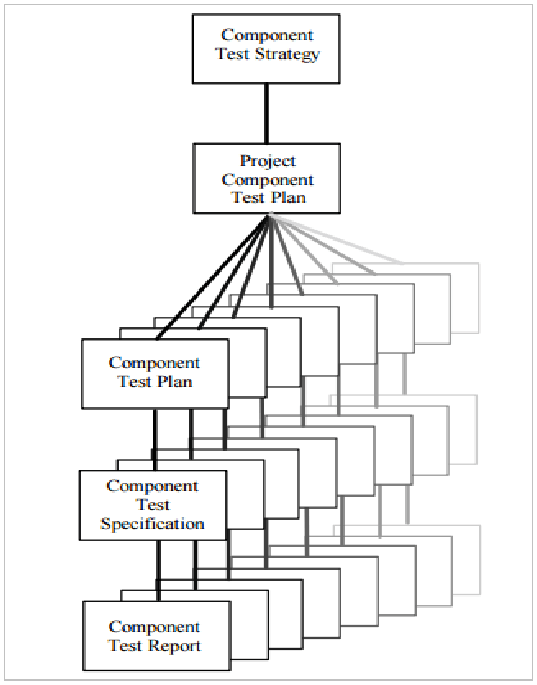
Test Approach:

- **Test-First/Test-Driven approach** – create the tests to drive the design and code construction!
- Instead of creating a design to tell you how to structure your code, you create a test that defines how a small part of the system should function.

Three steps:

1. Design test that defines how you think a small part of the software should behave (Incremental development).
2. Make the test run as easily and quickly as you can. Don't be concerned about the design of code, just get it to work!
3. Clean up the code. Now that the code is working correctly, take a step back and refactor to remove any duplication or any other problems that were introduced to get the test to run.

- There is a test strategy and test plan for component testing. And, for each component, there is a test scenario which will be further broken down in test cases. The below diagram represents the same:



- **The objective of Component Testing**

The main objective of component testing is to verify the input/output behavior of the test object. It ensures that the test object's functionality is working correctly and completely fine as per the desired specification.

The four major inputs to component level testing are:

- Project Test Plan
- System Requirements
- Component Specifications
- Component Implementations

Below we look at some of what extreme programming brings to the world of unit testing:

- Tests are written before the code
- Rely heavily on testing frameworks
- All classes in the applications are tested
- Quick and easy integration is made possible

10) What is functional system testing?

- **Functional System Testing : A requirement that specifies a function that a system or system component must perform**
- A Requirement may exist as a text document and/or a model
- **There is two types of Test Approach**
  - Requirement Based Functional Testing
  - Process Based Testing
- **Functional System Testing Functionality As below:**
- **Accuracy -** Provision of right or agreed results or effects
- **Interoperability -** Ability to interact with specified systems
- **Compliance -** Adhere to applicable standards, conventions, regulations or laws
- **Auditability -** Ability to provide adequate and accurate audit data
- **Suitability -** Presence and appropriateness of functions for specified tasks

11) What is Non-Functional Testing?

- Testing the attributes of a component or system that do not relate to functionality, e.g. reliability,efficiency, usability, interoperability, maintainability and portability
- Measuring the **characteristics** of the system/software that can be quantified on a varying scale- e.g. performance test scaling
- Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.

- It is the testing of “**how**” the system works. Non-functional testing may be performed at all test levels.

- The term non-functional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale, such as response times for performance testing.

- To address this issue, **performance testing is carried out to check & fine tune system response times.** The goal of performance testing is to reduce response time to an acceptable level

- Hence **load testing is carried out to check systems performance at different loads i.e. number of users accessing the system**

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

12) What is GUI Testing?

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

WHAT DO YOU CHECK IN GUI TESTING?

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

- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Approach of GUI Testing -

- **MANUAL BASED TESTING**
- Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.
- **RECORD AND REPLAY**
- GUI testing can be done using automation tools. This is done in 2 parts. During Record , test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.
- **MODEL BASED TESTING**
- A model is a graphical description of system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.

13) What is Adhoc testing?

- Adhoc testing is an informal testing type with an **aim to break the system**.
- It does not follow any test design techniques to create test cases.
- **In fact is does not create test cases altogether!**
- This testing is primarily performed **if the knowledge of testers in the system under test is very high**.
- Testers randomly test the application without any test cases or any business requirement document.
- Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.
- **Main aim of this testing is to find defects by random checking.**
- **Adhoc testing can be achieved with the testing technique called Error Guessing.**
- Error guessing can be done by the people having enough experience on the system to "**guess**" the most likely source of errors.
- **The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.**
- Some people seem to be naturally good at testing and others are good testers because they have a **lot of experience** either as a **tester or working with a particular system** and so are able to find out its weaknesses.
- This is why an error guessing approach, used after **more formal techniques** have been applied to some extent, can be very effective.
- It also saves a lot of time because of the assumptions and guessing made by the experienced testers to find out the defects which otherwise won't be able to find.
- **Using experience to postulate errors.**
- **Use Error Guessing to Complement Test Design Techniques.**

Types of Adhoc Testing

There are different types of Adhoc testing and they are listed as below:

1. Buddy Testing

- Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team. Buddy testing helps the testers develop better test cases and development team can also make design changes early. This testing usually happens after unit testing completion.

2. Pair testing

- Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing.

3. Monkey Testing

- Randomly test the product or application without test cases **with a goal to** break the system.

14) What is load testing?

- Load testing is **the process of subjecting a computer, peripheral, server, network or application to a work level approaching the limits of its specifications**.
- Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.
- Load testing is a kind of performance testing which determines a system's performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.

This testing usually identifies –

- The maximum operating capacity of an application.
- Determine whether current infrastructure is sufficient to run the application
- Sustainability of application with respect to peak user load
- Number of concurrent users that an application can support, and scalability to allow more users to access it.
- It is a type of non-functional testing. Load testing is commonly used for the
- Client/Server, Web based applications – both Intranet and Internet.

Need For Load Testing-

- Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to ensure optimal system performance, when that marketing brings in traffic.
- **For Example**
- Popular toy store **Toysrus.com**, could not handle the increased traffic generated by their advertising campaign resulting in loss of both marketing dollars, and potential toy sales.
- An Airline website was not able to handle 10000+ users during a festival offer.

Strategies of Load Testing-

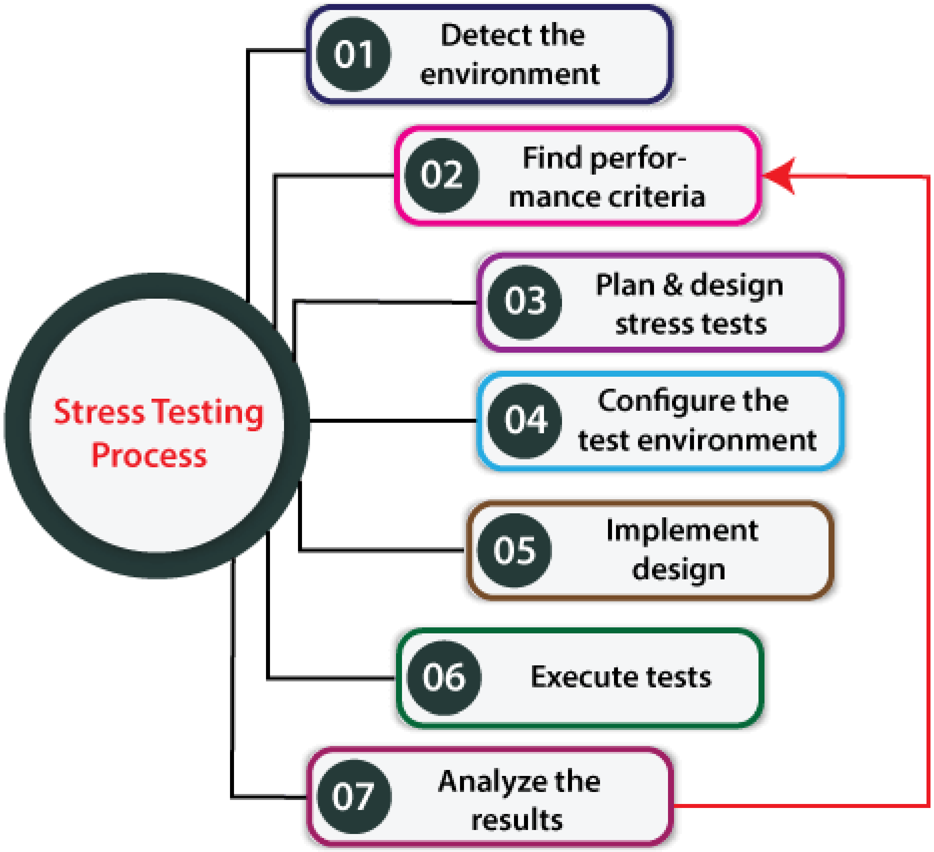
- Manual Load Testing
- In house(Organization) developed load testing tools
- Open source load testing tools
- Enterprise(Record and Play) load testing tools

Load Testing Tools –

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

15) What is stress Testing?

- System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.


Stress Testing Process
01 Detect the environment
02 Find perfor-mance criteria
03 Plan & design stress tests
04 Configure the test environment
05 Implement design
06 Execute tests
07 Analyze the results

- Stress testing is used to test the stability & reliability of the system.This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
- It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
- Stress Testing is done to make sure that the system would not crash under crunch situations.
- **Stress testing is also known as endurance testing.**

Types of Stress Testing –

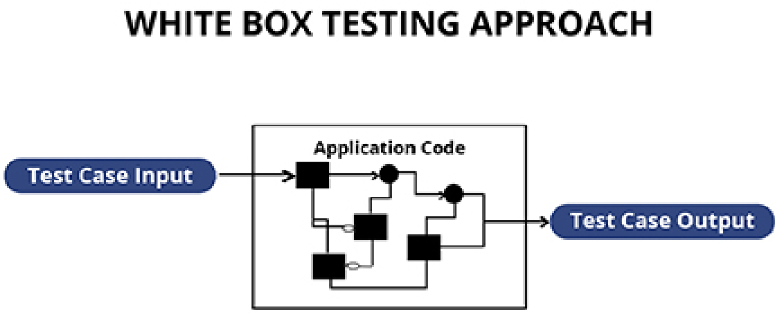
- **Application Stress Testing:**
- **Transactional Stress Testing:**
- **Systemic Stress Testing:**
- **Exploratory Stress Testing:**

Stress Testing Tools –

- **Stress**
- **Tester**
- **Neo Load**
- **App Perfect**

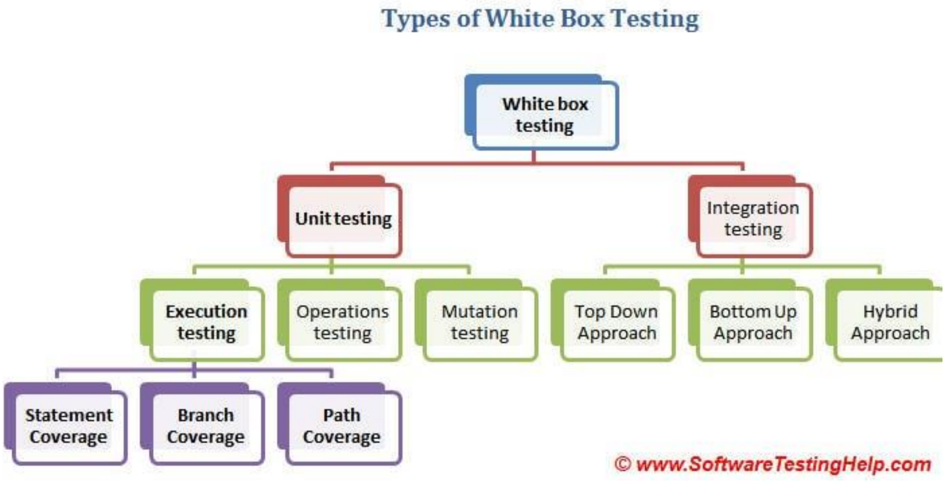
16) What is white box testing and list the types of white box testing?

- White box testing is **an approach that allows testers to inspect and verify the inner workings of a software system**—its code, infrastructure, and integrations with external systems.



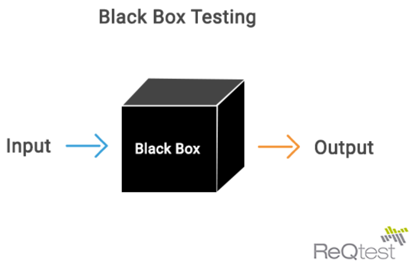
-
- **Structure-based testing** technique is also known as **'white-box'** or **'glass-box'** testing technique because here **the testers require knowledge of how the software is implemented, how it works**.
- Testing based upon the structure of the code
- Typically undertaken at Component and Component Integration Test phases by development teams

- White box testing is the detailed investigation of internal logic and structure of the code.
- White box testing is also called **glass testing or open box testing**. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
- **Types of White box Testing-**
  1. Statement coverage
  2. Decision coverage
  3. Condition coverage



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

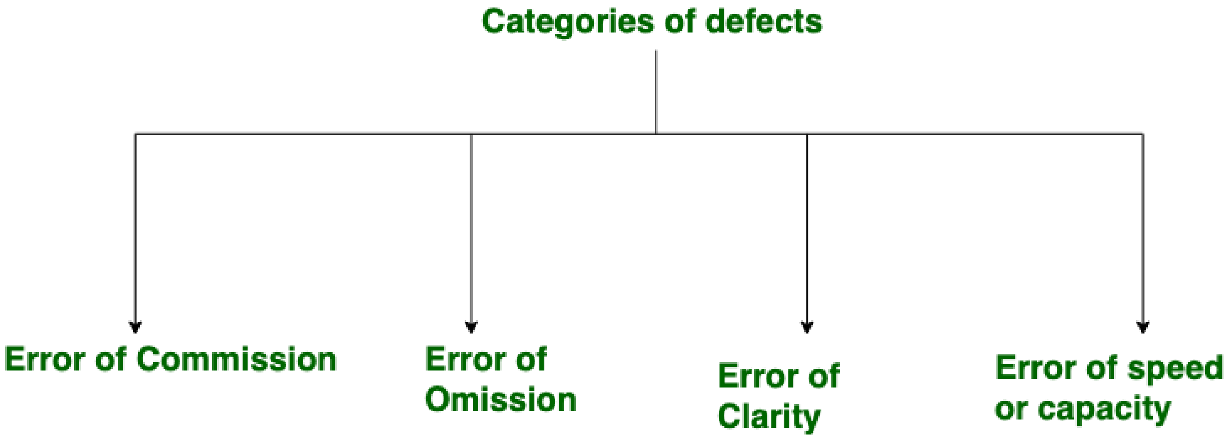
- **Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.**
- The testers **have no knowledge of how the system or component is structured inside the box**. In black-box testing the tester is concentrating on what the software does, not how it does it.
- The technique of testing without having **any knowledge of the interior workings of the application** is Black Box testing.



Techniques of Black Box Testing –

- There are four specification-based or black-box technique:
  - Equivalence partitioning
  - Boundary value analysis
  - Decision tables
  - State transition testing
  - Use-case Testing
  - Other Black Box Testing
    - Syntax or Pattern Testing

18) Mention what are the categories of defects?

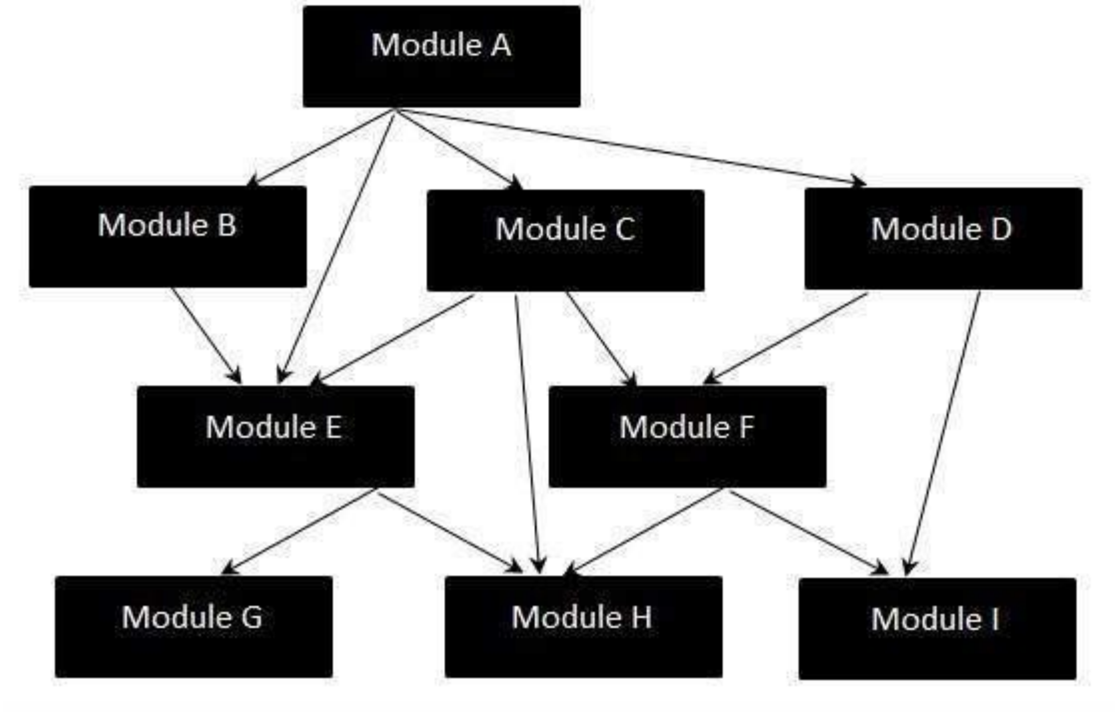


- **Error of Commission:**
- **Errors of Omissions:**
- **Error of Clarity:**
- **Error of Speed or Capacity:**

19) Mention what bigbang testing is?

- In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested as a whole.
- Big Bang testing has the advantage that everything is finished before integration testing starts.
- The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

Big Bang Integration - WorkFlow Diagram -



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.
- End of all testing – i.e. product Go Live
- End of phase of testing (e.g. hand over from System Test to UAT)
- Successful Testing of Integrated Application.
- Executed Test Cases are documented.
- All High prioritized bugs fixed and closed.
- Technical documents to be submitted followed by release Notes.

21) When should "Regression Testing" be performed?

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

Regression testing should be carried out:

- when the system is stable and the system or the environment changes
- when testing bug-fix releases as part of the maintenance phase
- It should be applied at all Test Levels

- It should be considered complete when agreed completion criteria for regression testing have been met
- Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation

Need of Regression Testing-

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

22) What is 7 key principles? Explain in detail?

1. Testing shows presence of Defects
2. Exhaustive Testing is Impossible!
3. Early Testing
4. Defect Clustering
5. The Pesticide Paradox
6. Testing is Context Dependent
7. Absence of Errors Fallacy

- **Testing shows presence of Defects-**
  - Testing can show that defects are present, but cannot prove that there are no defects.
  - 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.
  - As we find more defects, the **probability of undiscovered defects** remaining in a system reduces.
  - However Testing **cannot prove** that there are **no** defects present.

- **Exhaustive Testing is Impossible!-**
  - Testing everything including **all combinations of inputs and preconditions is not possible**.
  - So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
  - For example: In an application in **one screen there are 15 input fields**, each having 5 possible values, then to test all the valid combinations you would need **30 517 578 125 (515) tests**.
  - This is very unlikely that the project timescales would allow for this number of tests.
  - So, accessing and managing risk is one of the most important activities and reason for testing in any project.
  - We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions).

  - That is we must **Prioritiese** our testing effort using a **Risk Based Approach**.

- **Early Testing-**
  - Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
  - Testing activities should start as **early** as possible in the development life cycle
  - These activities should be focused on defined objectives – outlined in the Test Strategy
  - Remember from our Definition of Testing, that Testing doesn't start once the code has been written!

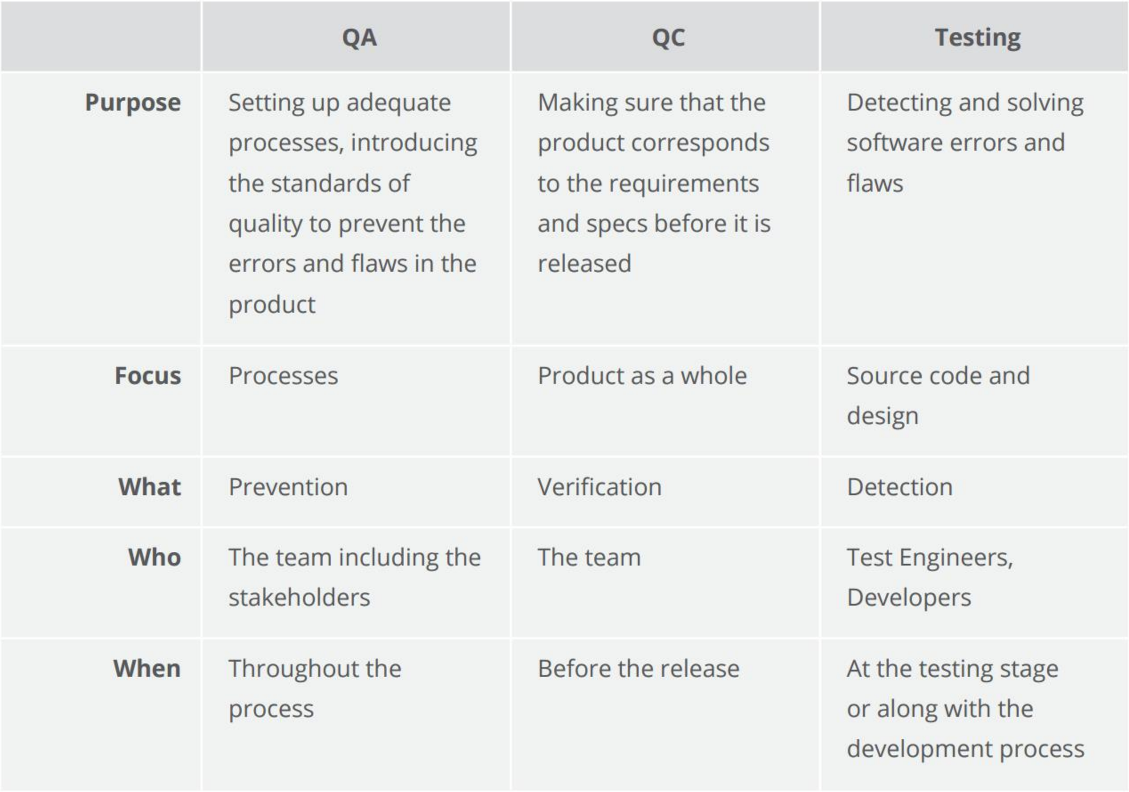
- **Defect Clustering-**
  - A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures.
  - Defects are not evenly spread in a system
  - They are 'clustered'
  - In other words, most defects found during testing are usually confined to a small number of modules
  - Similarly, most operational failures of a system are usually confined to a small number of modules
  - An important consideration in test prioritization!

- **The Pesticide Paradox-**
  - If the same tests are repeated over and over again, eventually the same set of **test cases will no longer find any new defects**.
  - To overcome this "pesticide paradox", the test cases need to be **regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.**
  - Testing identifies bugs, and programmers respond to fix them
  - As bugs are eliminated by the programmers, the software improves
  - As software improves the effectiveness of previous tests erodes

- **Testing is Context Dependent-**
  - Testing is done differently in different contexts
  - **Different kinds of sites are tested differently.**
  - For example
  - **Safety – critical software is tested differently from an e-commerce site.**
  - Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing
  - Also different industries impose different testing standards

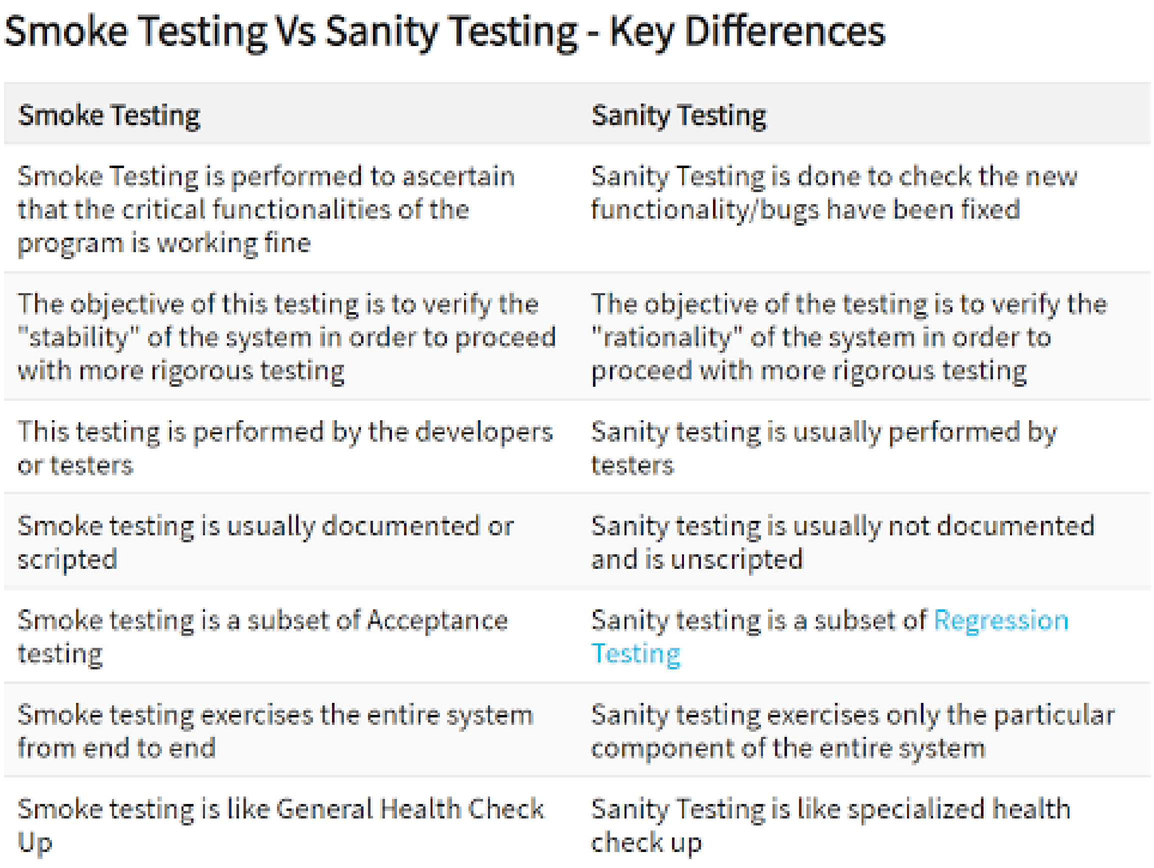
- **Absence of Errors Fallacy-**
  - If the system built is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help.
  - If we build a system and, in doing so, find and fix defects....
  - It doesn’t make it a **good** system
  - Even after defects have been resolved it may still be **unusable** and/or does not fulfil the users’ **needs and expectations**

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



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

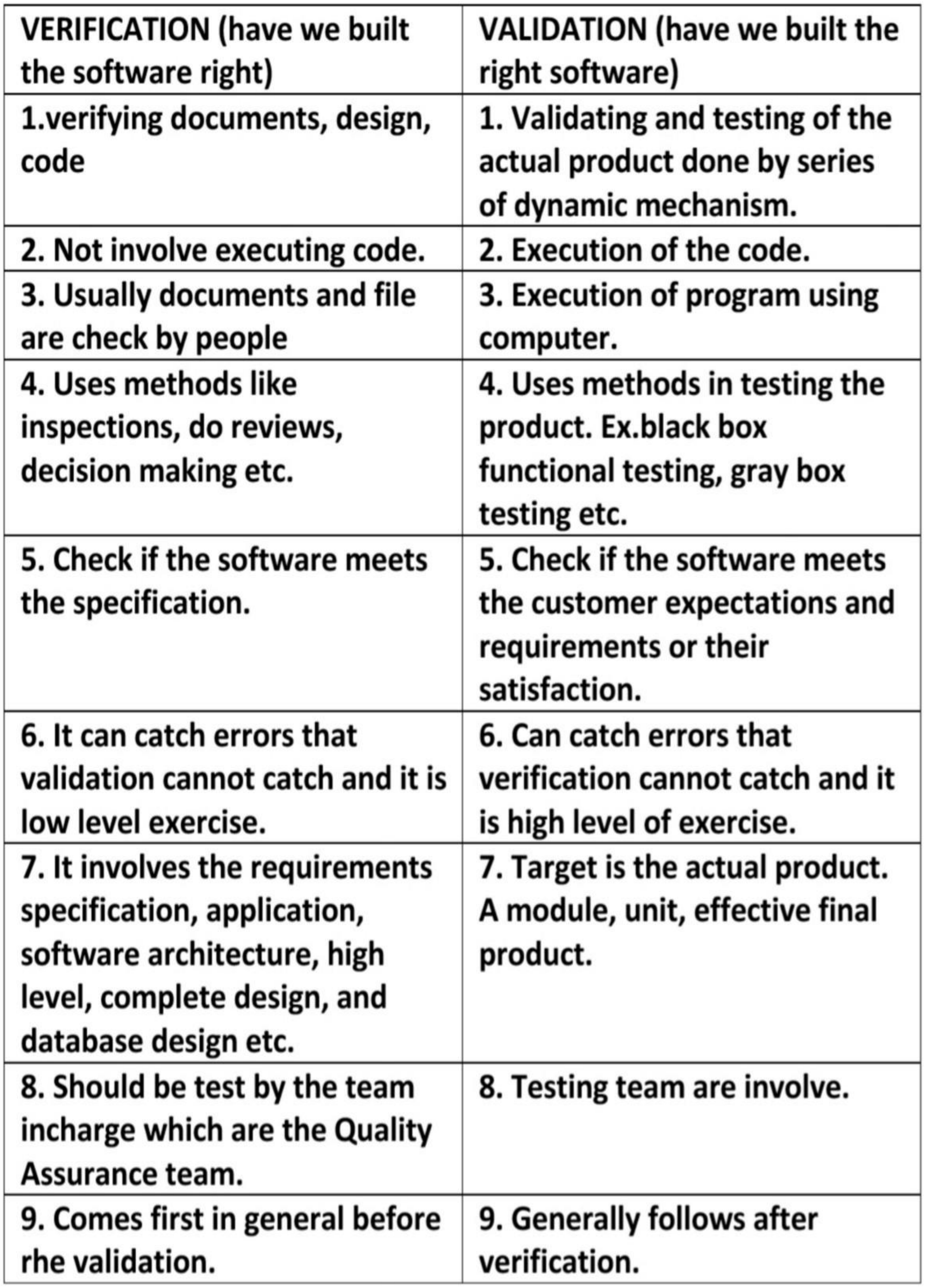
24) Difference between Smoke and Sanity?



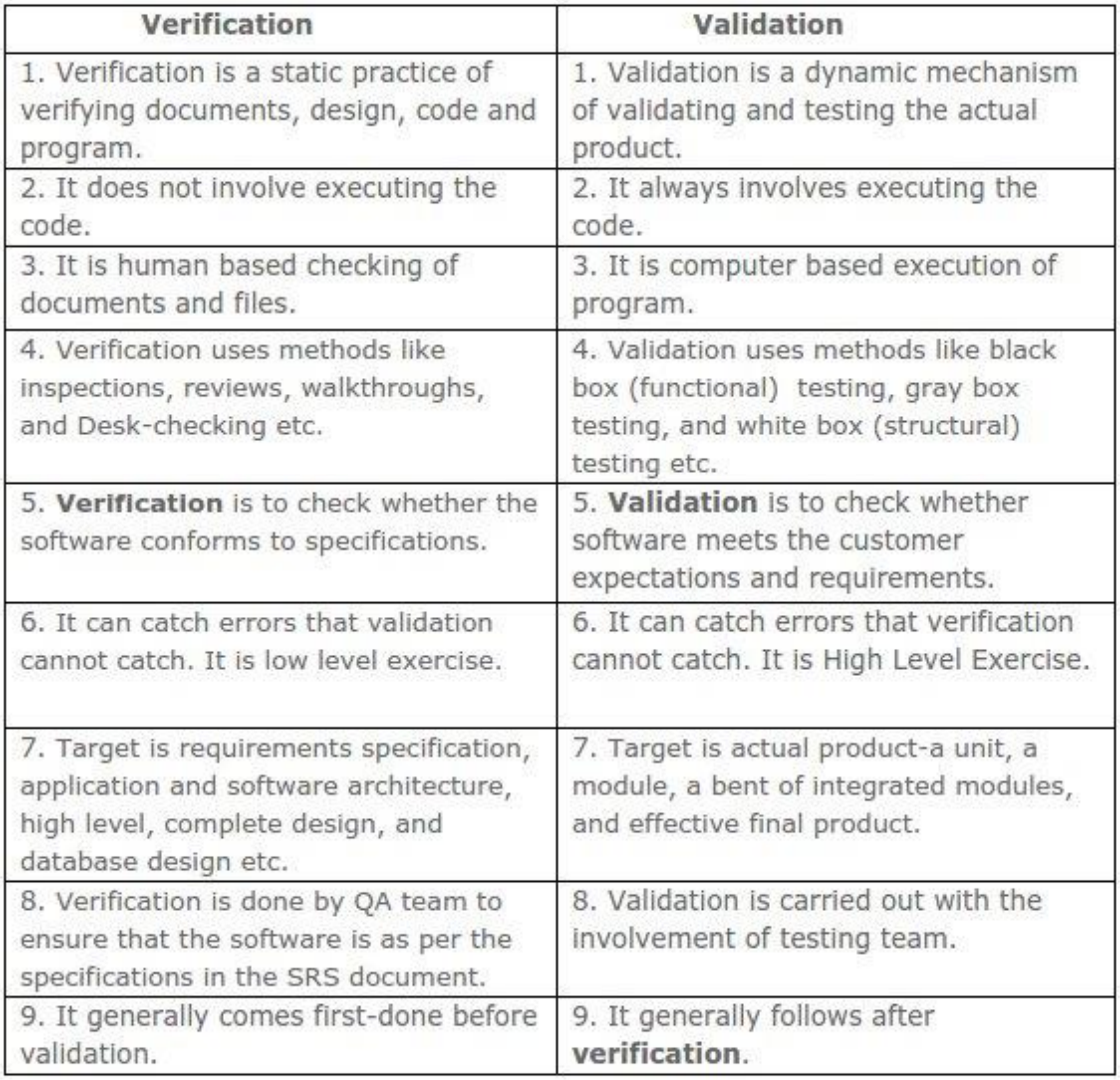
Smoke Testing Vs Sanity Testing - Key Differences

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 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
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 (have we built the software right)	VALIDATION (have we built the right software)
1.verifying documents, design, code	1. Validating and testing of the actual product done by series of dynamic mechanism.
2. Not involve executing code.	2. Execution of the code.
3. Usually documents and file are check by people	3. Execution of program using computer.
4. Uses methods like inspections, do reviews, decision making etc.	4. Uses methods in testing the product. Ex.black box functional testing, gray box testing etc.
5. Check if the software meets the specification.	5. Check if the software meets the customer expectations and requirements or their satisfaction.
6. It can catch errors that validation cannot catch and it is low level exercise.	6. Can catch errors that verification cannot catch and it is high level of exercise.
7. It involves the requirements specification, application, software architecture, high level, complete design, and database design etc.	7. Target is the actual product. A module, unit, effective final product.
8. Should be test by the team incharge which are the Quality Assurance team.	8. Testing team are involve.
9. Comes first in general before rhe validation.	9. Generally follows after verification.



Verification	Validation
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. **Verification** is to check whether the software conforms to specifications.	5. **Validation** is to check whether software meets the customer expectations and requirements.
6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after **verification**.

26) Explain types of Performance testing.

- Software performance testing is a means of quality assurance (QA). It involves testing software applications to ensure they will perform well under their expected workload.

The focus of Performance testing is checking a software programs

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

Types of Performance Testing

- **Load testing**
- **Stress testing**
- **Endurance testing**
- **Spike testing**
- **Volume testing**
- **Scalability 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.

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

- "A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then it is called bug, build does not meet the requirements then it is failure".

What is an error?

- We can't compile or run a program due to coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an **error**.

What is a defect?

- The variation between the actual results and expected results is known as defect.
- If a developer finds an issue and corrects it by himself in the development phase then it's called a defect.

What is a bug?

- If testers find any mismatch in the application/system in testing phase then they call it as Bug.
- As I mentioned earlier, there is a contradiction in the usage of Bug and Defect. People widely say the bug is an informal name for the defect.

What is a failure?

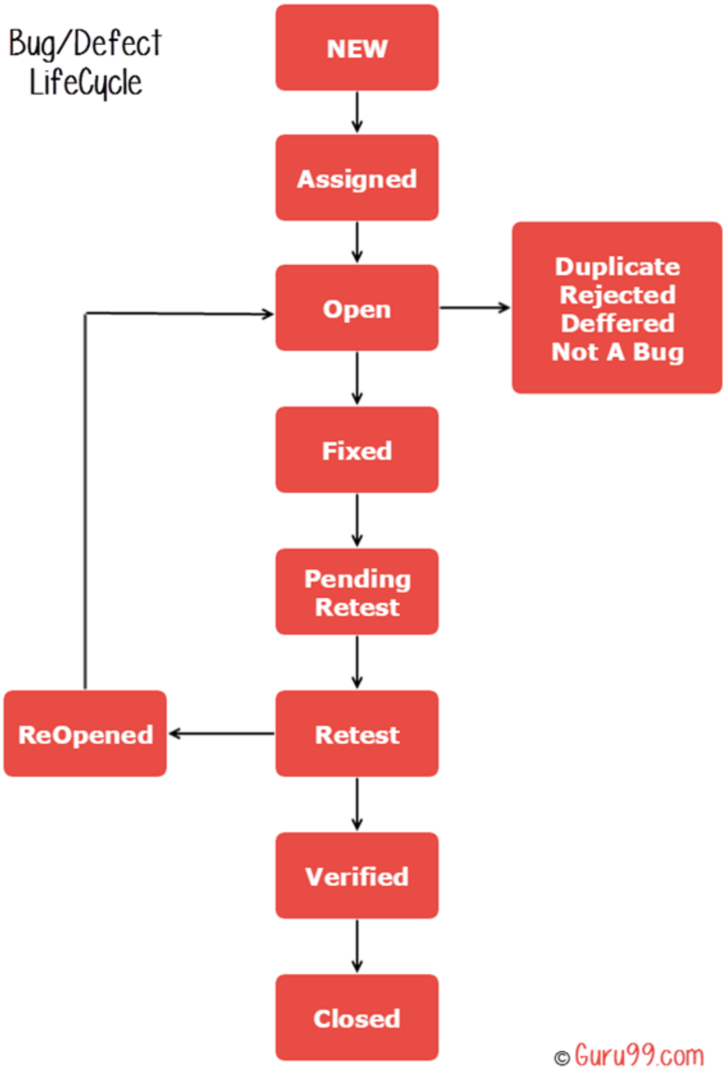
- Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as **failure.**

28) Difference between Priority and Severity

Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.
Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.
Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.
Who Decides the Defect	The testing engineer basically decides a defect’s severity level.	The product manager basically decides a defect’s priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

29) What is Bug Life Cycle?

- In Software Development process, Defect Life Cycle is **life cycle of defect or bug from which it goes through covering the specific set of states in its entire life**. Mainly bug life cycle refers to its entire states starting from a new defect is detected to closing of that defect by tester.
- “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”
- The duration or time span between the first time defects is found and the time that it is closed successfully, rejected, postponed or deferred is called as ‘Defect Life Cycle’.



30) Explain the difference between Functional testing and Nonfunctional testing

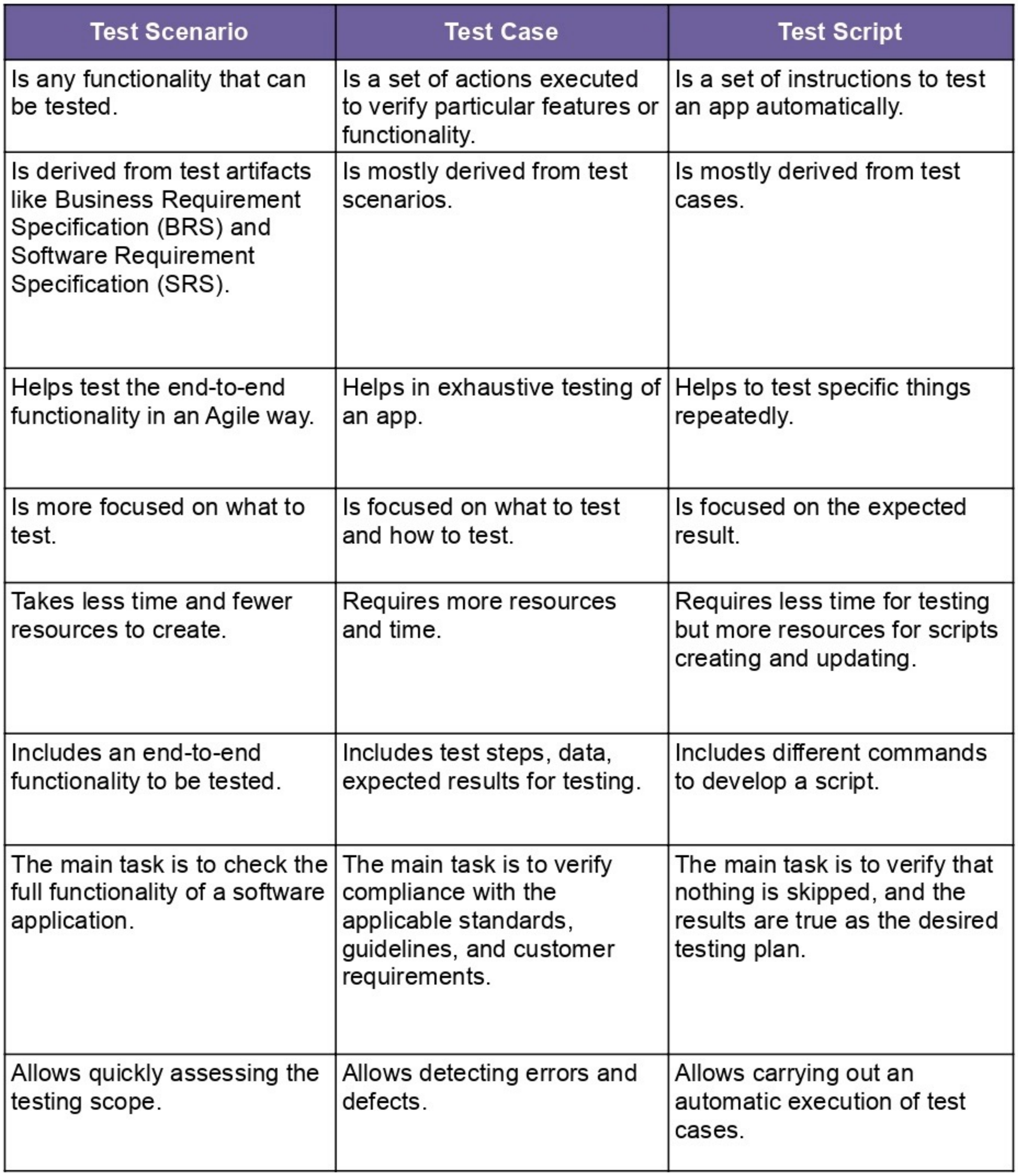
Functional Testing	Non-functional Testing
It verifies the operations and actions of an application.	It verifies the behavior of an application.
It is based on requirements of customer.	It is based on expectations of customer.
It helps to enhance the behavior of the application.	It helps to improve the 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)?

Criterion	**SDLC**	**STLC**
Origin	Development Life Cycle	Testing Life Cycle
Stands for	SDLC stands for Software Development Life Cycle	STLC stands for Software Testing Life Cycle
Definition	(SDLC) aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.	Software Testing Life Cycle (STLC) identifies what test activities to carry out and when to accomplish those test activities. Even though testing differs between Organizations, there is a testing life cycle.
Focus	On both development and testing process	On only testing process
Relationship	It is taken as the predecessor	It is taken as the successor
Phases	Requirement Gathering, Analysis, Design, Coding, Testing, Deployment & maintenance	Requirement Analysis, Test Planning, Test Design, Environment Setup, Test Execution, Test Closure
Requirement Gathering Phase	Business analyst gathers the requirements and create Development Plan	QA team analyses requirement documents and create System Test Plan
Design Phase	The development team develops the high and low-level design of the software based on the requirements	Test Architect or a Test Lead usually plan the test Strategy
Coding Phase	The actual code is developed as per the designed document	The QA team prepares the test Environment
Testing Phase	Actual testing is done in this phase. It includes Unit, Integration, System, Retesting & Regression testing etc., Also the development team involves in fixing the bugs reported	Actual testing is done in this phase. Defect reporting & retesting is done here
Deployment or Maintenance Phase	The development team involves in support and release updates	The QA team executes regression suites to check maintenance code deployed

When it is performed	The SDLC phases are performed before the STLC phases	The STLC phases are performed after the SDLC Phases
Outcome	A good quality software product	A bug free software

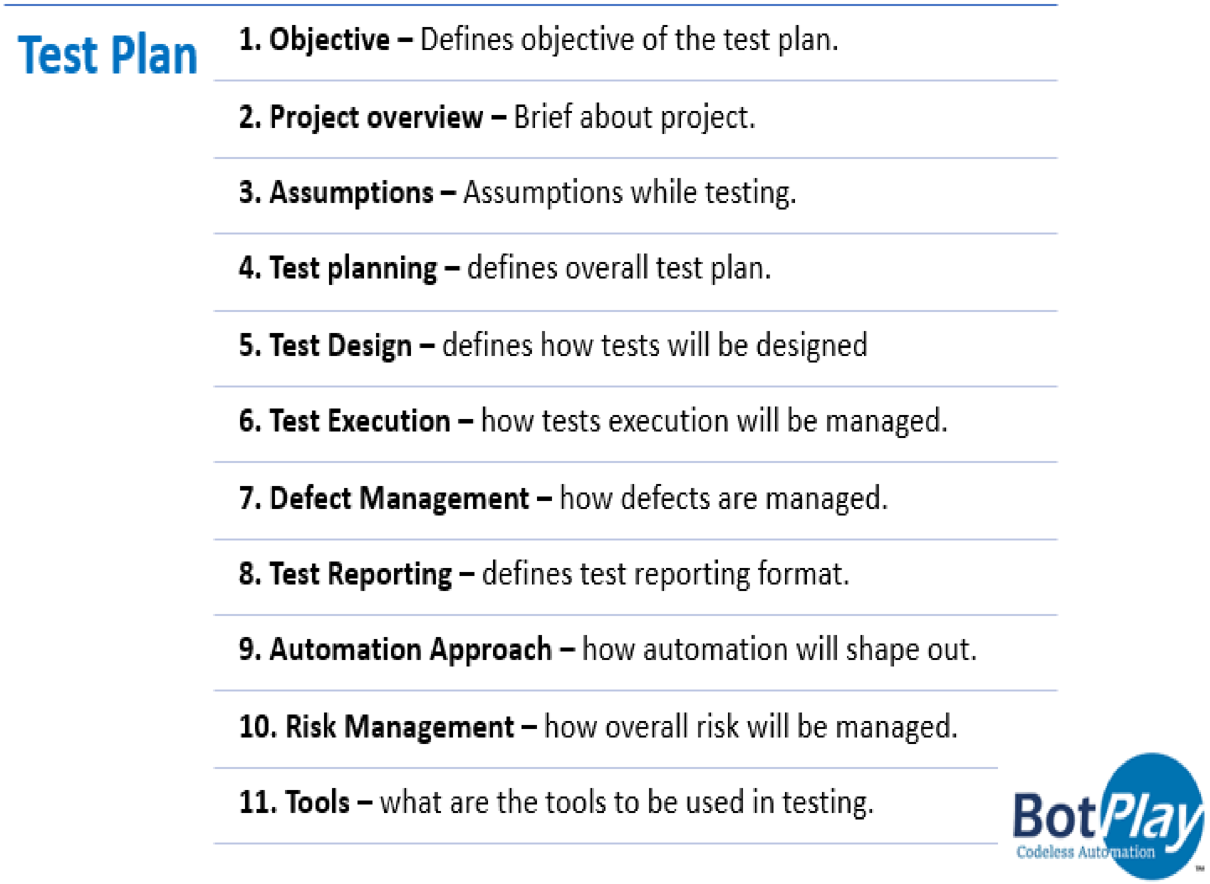
32) What is the difference between test scenarios, test cases, and test script?



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.

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

- A test plan is **a detailed document which describes software testing areas and activities**. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables. The test plan is a base of every software's



Test Plan

1. **Objective –** Defines objective of the test plan.
2. **Project overview –** Brief about project.
3. **Assumptions –** Assumptions while testing.
4. **Test planning –** defines overall test plan.
5. **Test Design –** defines how tests will be designed
6. **Test Execution –** how tests execution will be managed.
7. **Defect Management –** how defects are managed.
8. **Test Reporting –** defines test reporting format.
9. **Automation Approach –** how automation will shape out.
10. **Risk Management –** how overall risk will be managed.
11. **Tools –** what are the tools to be used in testing.

BotPlay Codeless Automation

34) What is priority?

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

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

35) What is severity?

- **Severity is absolute and Customer-Focused.** It is the extent to which the defect can affect the software. In other words it defines the impact that a given defect has on the system.
- **For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.
- **Severity can be of following types:**
- **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
- **Major (High):** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
- **Moderate (Medium):** The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
- **Minor (Low):** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
- **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

36) Bug categories are…

- Functional errors.
- Syntax errors.
- Logic errors.
- Calculation errors.

- Unit-level bugs.
- System-level integration bugs.
- Out of bounds bugs.

37) Advantage of Bugzilla.

- Open source, free bug tracking tool.
- Automatic Duplicate Bug Detection.
- Search option with advanced features.
- File/Modify Bugs By Email.
- Move Bugs between Installs.
- Multiple Authentication Methods (LDAP, Apache server).
- Time Tracking.
- Automated bug reporting; has an API to interact with system.
- Integrated email capabilities.
- Detailed permissions system.
- Optimized database structure to enhance performance.
- Robust security.
- Powerful query tool.
- Ideal for small projects.

38) Difference between priority and severity.

Parameters	**Severity in Testing**	**Priority in Testing**
Definition	Severity is a term that denotes how severely a defect can affect the functionality of the software.	Priority is a term that defines how fast we need to fix a defect.
Parameter	Severity is basically a parameter that denotes the total impact of a given defect on any software.	Priority is basically a parameter that decides the order in which we should fix the defects.
Relation	Severity relates to the standards of quality.	Priority relates to the scheduling of defects to resolve them in software.
Value	The value of severity is objective.	The value of priority is subjective.
Change of Value	The value of Severity changes continually from time to time.	The value of Priority changes from time to time.

Who Decides the Defect	The testing engineer basically decides a defect’s severity level.	The product manager basically decides a defect’s priority level.
Types	There are 5 types of Severities: Cosmetic, Minor, Moderate, Major, and Critical.	There are 3 types of Priorities: High, Medium, and Low.

39) What are the different Methodologies in Agile Development Model?

- **Kanban:**

  Kanban is a simple, visual means of managing projects that enables teams to see the progress so far and what’s coming up next. Kanban projects are primarily managed through a Kanban board, which segments tasks into three columns: “To Do,” “Doing,” and “Done.”

- **Scrum:**

  Scrum is similar to Kanban in many ways. Scrum typically uses a Scrum board, similar to a Kanban board, and groups tasks into columns based on progress. Unlike Kanban, Scrum focuses on breaking a project down into sprints and only planning and managing one sprint at a time. Scrum also has unique project roles: Scrum master and product owner.

- **Extreme Programming (XP):**

  Extreme Programming (XP) was designed for Agile software development projects. It focuses on continuous development and customer delivery and uses intervals or sprints, similar to a Scrum methodology. However, XP also has 12 supporting processes specific to the world of software development:

- Planning game
- Small releases
- Customer acceptance tests
- Simple design
- Pair programming
- Test-driven development

- Refactoring
- Continuous integration
- Collective code ownership
- Coding standards
- Metaphor
- Sustainable pace
- **Feature-driven development (FDD):**

Feature-driven development is another software-specific Agile framework. This methodology involves creating software models every two weeks and requires a development and design plan for every model feature. It has more rigorous documentation requirements than XP, so it's better for teams with advanced design and planning abilities. FDD breaks projects down into five basic activities:

- Develop an overall model
- Build a feature list
- Plan by feature
- Design by feature
- Build by feature
- **Dynamic Systems Development Method (DSDM):**

The Dynamic Systems Development Method (DSDM) was born of the need for a common industry framework for rapid software delivery. Rework is to be expected, and any development changes that occur must be reversible. Like Scrum, XP, and FDD, DSDM uses sprints. This framework is based on eight fundamental principles:

- Focus on the business need
- Deliver on time
- Collaborate
- Never compromise quality
- Build incrementally from firm foundations
- Develop iteratively
- Communicate continuously and clearly
- Demonstrate control
- **Crystal**:

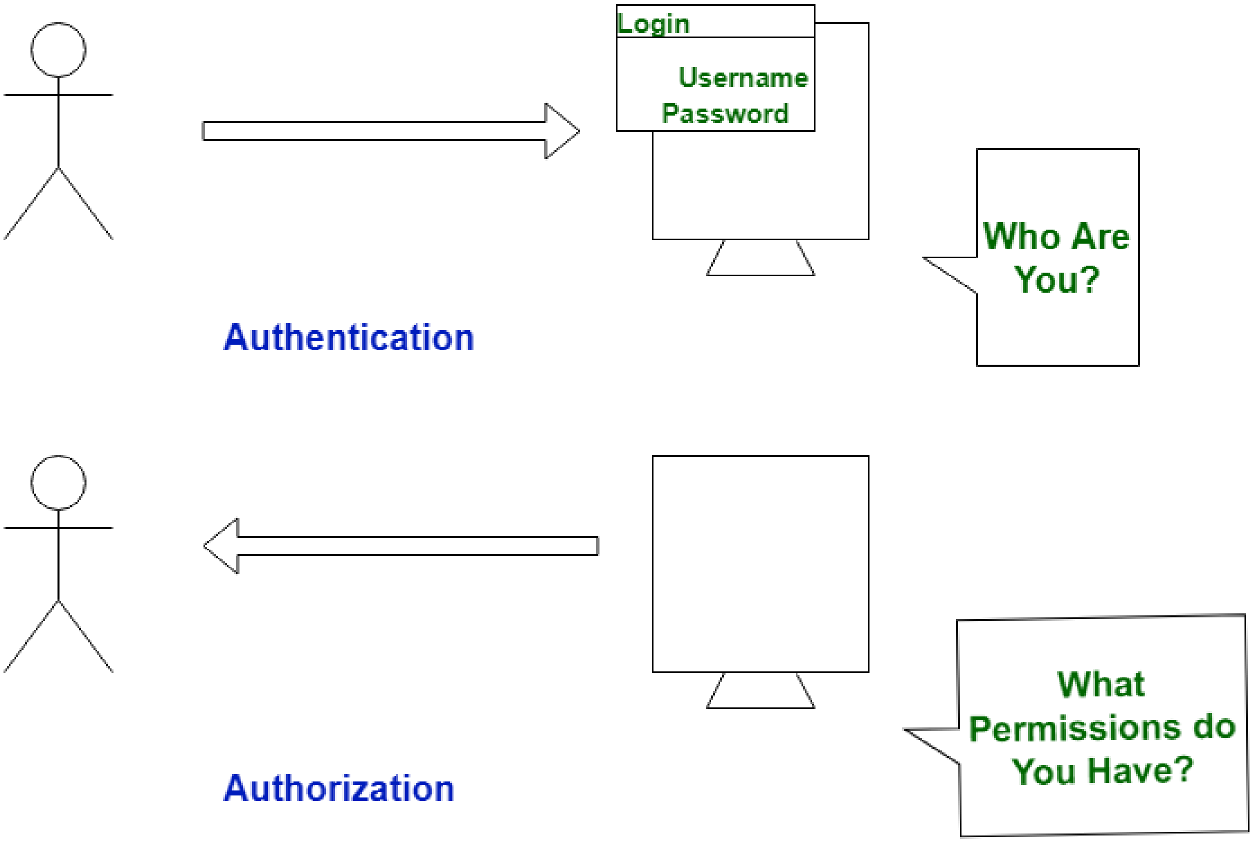
  Crystal is a family of Agile methodologies that includes Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, etc. Each has a unique framework. Your choice depends on several project factors, such as your team size, priorities, and project criticality.
- **Lean:**

  Lean development is often grouped with Agile, but it’s an entirely different methodology that happens to share many of the same values. The main principles of the Lean methodology include:

- Eliminating waste
- Build quality in
- Create knowledge
- Defer commitment

- Deliver fast
- Respect people
- Optimize the whole

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



Authentication	Authorization
In the authentication process, the identity of users are checked for providing the access to the system.	While in authorization process, a the person's or user's authorities are checked for accessing the resources.
In the authentication process, users or persons are verified.	While in this process, users or persons are validated.
It is done before the authorization process.	While this process is done after the authentication process.
It needs usually the user's login details.	While it needs the user's privilege or security levels.
Authentication determines whether the person is user or not.	While it determines **What permission does the user have?**
Generally, transmit information through an ID Token.	Generally, transmit information through an Access Token.
The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process.	The OAuth 2.0 protocol governs the overall system of user authorization process.
Popular Authentication Techniques- • Password-Based Authentication • Passwordless Authentication • 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) • Single sign-on (SSO) • Social authentication	Popular Authorization Techniques- • Role-Based Access Controls (RBAC) • SON web token (JWT) Authorization • SAML Authorization • OpenID Authorization • OAuth 2.0 Authorization

Authentication	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.

41) When to used Usability Testing?

- The goal of usability testing is **to understand how real users interact with your website** and make changes based on the results. It is important to be sure that your app or website is easy to navigate and that tasks can be completed with ease; otherwise, people will leave and go to a competitor's site.
- Aesthetics and design are important. How well a product looks usually determines how well it works.
- There are many software applications / websites, which miserably fail, once launched, due to following reasons –
- Where do I click next?
- Which page needs to be navigated?
- Which Icon or Jargon represents what?
- Error messages are not consistent or effectively displayed

- Session time not sufficient.
- Usability Testing identifies usability errors in the system early in development cycle and can save a product from failure.

42) What is the procedure for GUI Testing?

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

Approach of GUI Testing

- **MANUAL BASED TESTING**
  - Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.
- **RECORD AND REPLAY**
  - GUI testing can be done using automation tools. This is done in 2 parts. During Record , test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.
- **MODEL BASED TESTING**
  - A model is a graphical description of system's behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.

43) To create HLR & TestCase of Web Based (WhatsApp web , Instagram) 1. WhatsApp Web: https://web.whatsapp.com/

2.InstaGram Web : https://www.instagram.com/accounts/login/?hl=en

- https://github.com/rajani15st/rajani15novst.git

44) To create HLR and Test Case on this Link. https://artoftesting.com/

- https://github.com/rajani15st/rajani15novst.git

45) To create HLR & Test Case of 1)(Instagram , Facebook) only first page

2) Facebook Login Page: https://www.facebook.com/

- https://github.com/rajani15st/rajani15novst.git