Manual Testing

1. What is Testing / Software Testing?

Software testing is a method to check whether the software application matchesthe requirements and to ensure that the software is bug/defect free.

2. Seven Testing Principles

1. **Testing shows the presence of defects** (only while testing, we canidentify that the application has any errors.)

2. **Exhaustive Testing is not possible** (means testing or quality assurance approachin which all possible combinations of scenarios and use/test cases are used for testingis impossible.)

3. **Early Testing** (means that all the testing activities should start in the early stagesof the software development life cycles 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 development.)

4. **Defect Clustering** (Small number of modules contains most of the bugs detected or show the most operational failures. e.g.- 80% of the bug mostly found on20% of the modules.) This is also in accordance to Pareto's Principle.

5. **Pesticide Paradox**(If the same testing approaches, techniques or methods are repeated over and over again on a same test case, eventuallythat test case will no longer yield new bugs.)

6. **Testing is context-dependent** (To test different types of application, we will takethe help of various kinds of testing, different techniques, approaches, and multiple methods.)

7. **Absence of errors fallacy** (Means even if built software is 99% bug-free but itdoes not follow the user requirement then it is unusable.)

3. What is SDLC?

Software Development Life Cycle, is a process of developing the software which has various phases like

1.Requirement Phase (Business Analyst/ Project Manager gather all necessariesinformation from Client).

2. Design Phase (The designing of software, programming Language,

database,hardware, and software is taken in this phase).
3. Build/ Development Phase (Developers starts writing the codes for application)
4. Testing Phase (Testing the application as per requirements and if any bug foundthen assigned to the developer).
5. Deployment/Deliver Phase (After proper testing the application is handover tothe client or customer).
6. Maintenance Phase (We provide maintenance for the application for time totime in future reference).

4. What is STLC?

Software Testing Life Cycle, includes all the phases of testing process. Which are -

1. **Requirement Analysis** (Test team gathers SRD and plans all the necessary testsas per requirement).
2. **Test Plan Creation** (Test team plans the strategy, approach and cost of thetesting).
3. **Test Case development** (Test scenario and test cases are written by the tester asper requirement).
4. **Test Environment Setup** (Environment setup requires a group of essentialsoftware and hardware to create a test environment. This is done by the senior developer).
5. **Test Execution** (Test team do the real-time validation of product and findingbugs).
6. **Test Cycle Closure** (Once testing is completed, matrix, reports, results aredocumented).

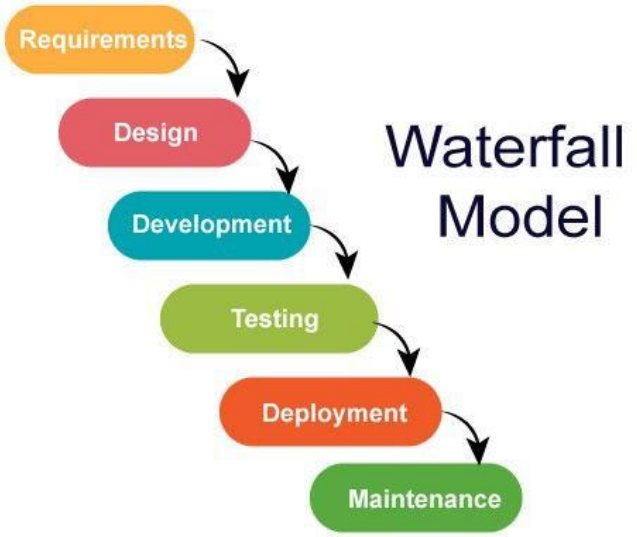
Types of Environment in Software Industry:

1. **Dev Environment** : The Development Team work on this environment.
2. **QA/Testing Environment** : The Testing Team performs the execution on this environment.
3. **Staging/Lab Environment**: After Testing, the build is kept on staging/lab environment. Here user Acceptance Testing is done before releasing the software.
4. **Production Environment**: This is the environment which the real user sees and uses. After all the testing, the build is deployed on production environment for the actual users.

5. SDLC Models:

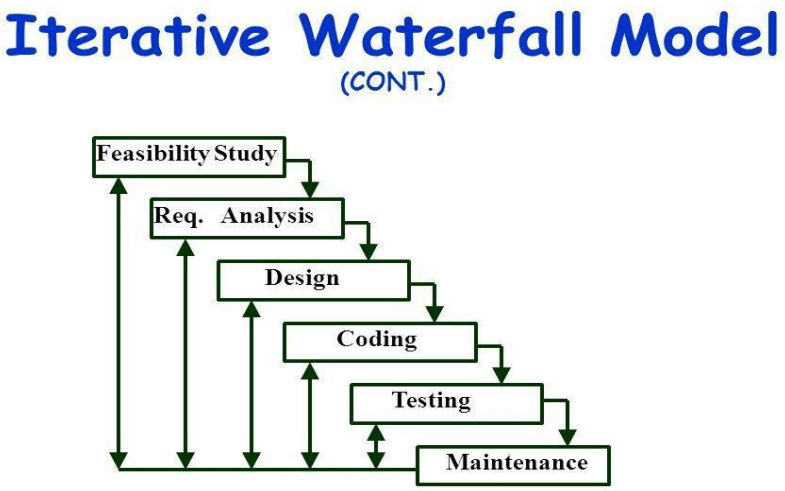
1. **Waterfall model:**

It is the first approach and the basic model used in software development. Then execution happens in the sequence order, which means that the outcome of the one- stage is equal to the input of another stage. That's why it is also known as the Linear- sequential life cycle model. To avoid the overlapping issues of the multiple phases, every stage should be completed before moving to the next stage(No back tracking isallowed)



2.Iterative model:

It is the modified version of waterfall model. Feedback can be sent to the previousphases (i.e. back tracking is possible). But no phase overlapping is allowed.



3.Spiral model:

It is the combination of waterfall and iterative model. It is used basically for highrisk project, which reduces the no. of risks and also any changes required can be done on later stages.

Spiral Process Model

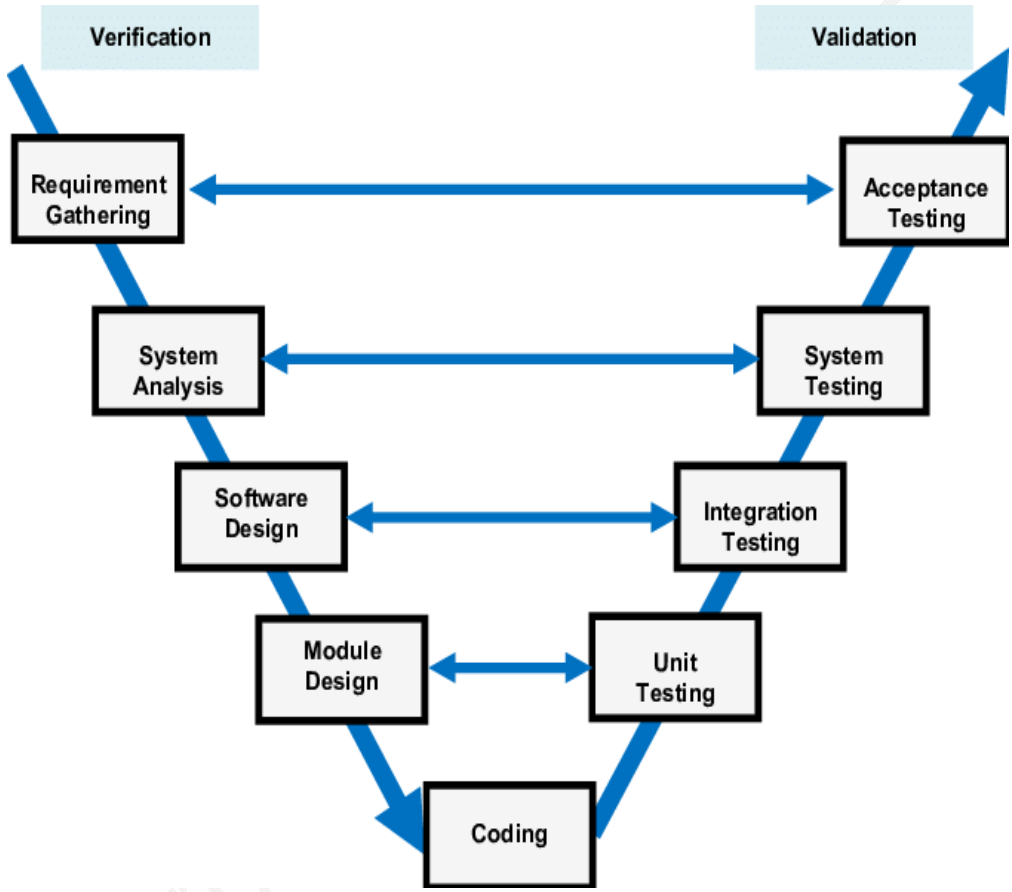
4. V- shaped model:

It is also known as verification and validation model, in this model testing phase isassociated with each corresponding development phase.

a) **Verification** is the static testing and **validation** is dynamic testing

b) **Verification** means are we building the software right? And **validation** means arewe building right software?

c) **Verification** does not include execution of code and **validation includes** executionof code.



5. Agile Model:

It is combination of iterative and incremental model. It focuses on adaptability and customer satisfaction by rapid delivery. In agile model product are divided into small incremental builds and these builds are provided in iteration which last for 1-3 weeks.Agile model is suitable for changing requirement and fast delivery.

6. Types of software testing:

(a)Manual Testing:

Manual Testing is a software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application.

(b)Automation Testing

Automation testing, which uses some specific tools to execute the test scripts without any human interference. It is the most accepted way to enhance the efficiency, productivity, and test coverage of Software testing.

With the help of an automation testing tool, we can easily fetch the test data, handle the test implementation, and compare the actual output against the expected outcome.

Automation Testing vs Manual Testing: Key Difference:-

- Manual Testing is done manually by QA analyst (Human) whereas Automation Testing is done with the use of script, code and automation tools (System software) by a tester.
- Manual Testing process is often accurate because of human ability to see and perceive if any other issues exist apart from normal test case flow whereas the Automation process is reliable only for a defined flow because it is code and script based.
- Manual Testing is a time-consuming process whereas Automation Testing is very fast.
- Manual Testing is possible without programming knowledge whereas Automation Testing is not possible without programming knowledge.
- Manual Testing allows random Testing whereas Automation Testing doesn't allow random Testing.

7. Types of Manual Testing:

(a)White box Testing

It can be done by developer (can be done by whit box tester too), where they check every line ofcode before deploying the code to the Testing Environment.

(b) Black box Testing- It is done by the test engineer where they can check thefunctionality of an application. (Code is not visible)

(c) Grey box Testing- It is combination of white box and black box testing. Grey box testing includes access to the internal code, so it is performed by a person who knows coding as well as testing. A Grey box tester not only reports the functionality failure but also the reason for the failure at code level.

8. Types of Black Box Testing:

(a)Functional Testing- Functional testing is a type of software testing in which the system is tested ***against the functional requirements and specifications***. Functional testing ensures that the requirements or specifications are properly satisfied by the application.

(b)Non-functional testing- Non-functional testing is a type of software testing that is performed to verify whether the ***behavior of the system*** is as per the requirement or not.

9. Difference between Functional and Non-Functional Testing:

Functional Testing	**Non-Functional Testing**
1. It verifies the operations and actions of *behavior* of anan application.	1. It verifies the application.
2. Functional testing is based on the business based requirement. requirement	2. Non- functional testing is on the Performance
3. For e.g. Unit Testing, Smoke Testing, Testing,Integration Testing, Regression Testing	3. For e.g., 1. Performance Load Testing, Stress Testing.

10. Types of Functional Testing:

(a)Unit Testing- Here, individual unit/ component of an application are tested. It isdone during the development phase.

For e.g., Like If Developer Created any website login page and developer will login with a correct password and a correct username and it will redirect the user to result in the page it is called successful unit testing.

(b) Integration Testing- Here, individual unit/component are combined or groupedtogether and is tested. For e.g., in Gmail if we send mail to anyone and verifyin the Send Items folder to check if the send mail is there.

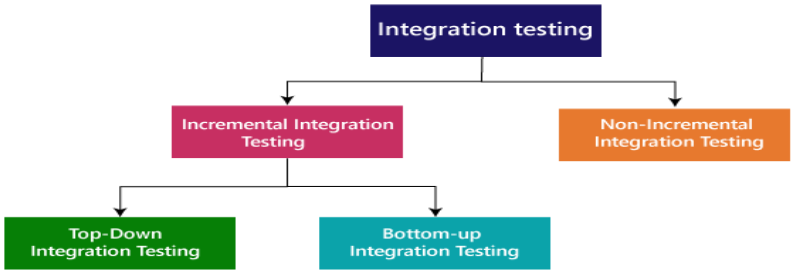
(c) System Testing- Here, complete integrated software is tested (End-to-End) to verify the customer requirement. Application as a ***whole*** is tested using ***functional and non-functional testing***. It is carried out by the QA team after the completion of integrationtesting and before the final acceptance testing.

(d) User-acceptance Testing- It is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional,integration and system testing is done.

11. Types of Integration testing:

Integration testing can be classified into two parts:

- **Incremental integration testing**
- **Non-incremental integration testing**



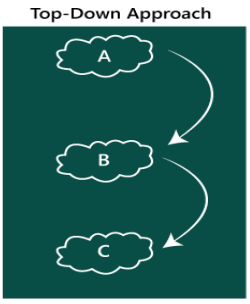
In the **Incremental Testing** approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application. Then the other related modules are integrated incrementally and the process continues until all the logically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

- Bottom Up
- Top Down

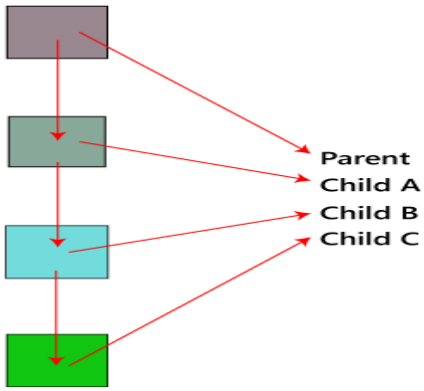
Top-Down Approach

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.

In this testing ***Stubs are used as temporary modules***, if a module is not ready forintegration testing.



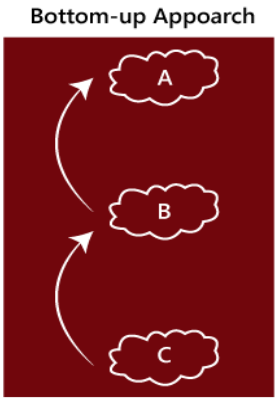
In the top-down approach, we will be ensuring that the module we are adding is the **child of the previous one like Child C is a child of Child B** and so on as we can see in the below image:



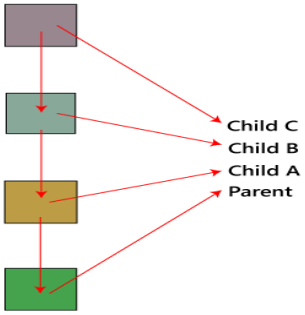
Bottom-Up Method

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.

In this ***Driver are used as temporary modules.***



In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one** as we can see in the below image:



Non- incremental integration testing

We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child.

Big Bang Method

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.

12. Non-Functional Testing

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing.

Type of Non-Functional Testing:

Performance Testing

Performance Testing eliminates the reason behind the slow and limited performance of the software. Reading speed of the software should be as fast as possible.

For Performance Testing, a well-structured and clear specification about expected speed must be defined. Otherwise, the outcome of the test (Success or Failure) will not be obvious.

While doing performance testing on the application, we will concentrate on the various factors like **Response time, Load, and Stability** of the application.

Response time: Response time is the time taken by the server to respond to the client's request.

Load: Here, Load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.

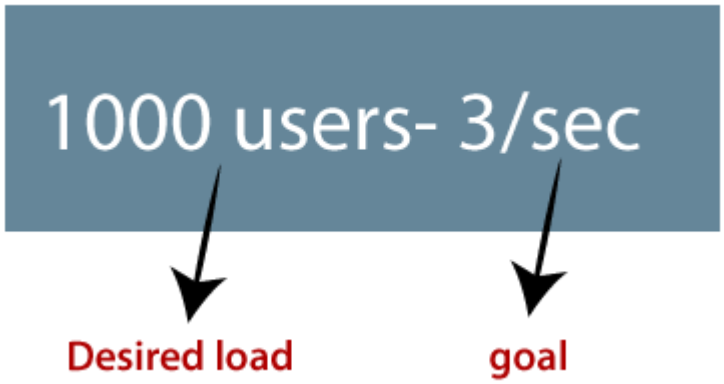
Stability: For the stability factor, we can say that, when N-number of users using the application simultaneously for a particular time.

Load Testing

Load testing involves testing the system's loading capacity. Loading capacity means more and more people can work on the system simultaneously.

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

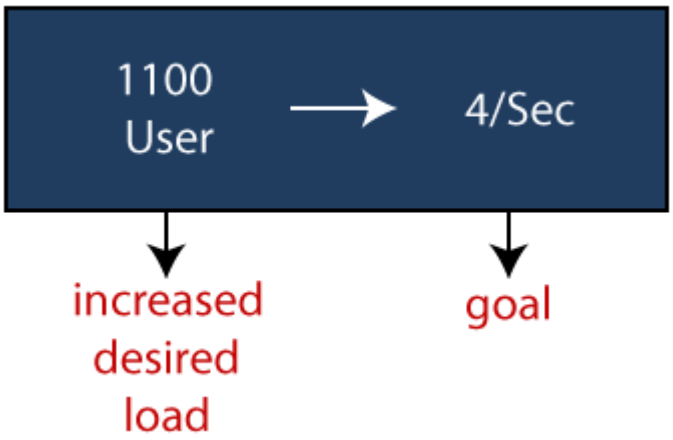
For example: In the below image, **1000 users** are the **desired load**, which is given by the customer, and **3/second** is the **goal** which we want to achieve while performing a load testing.



Stress Testing

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

For example: If we took the above example and increased the desired load 1000 to 1100 users, and the goal is 4/second. While performing the stress testing in this scenario, it will pass because the load is greater (100 up) than the actual desired load.



Scalability Testing

Checking the performance of an application by increasing or decreasing the load in particular scales (number of users) is known as **scalability testing**. Upward scalability and downward scalability testing are called scalability testing.

Scalability testing is divided into two parts which are as follows:

- **Upward scalability testing**
- **Downward scalability testing**

Upward scalability testing

It is testing where we **increase the number of users on a particular scale** until we get a crash point. We will use upward scalability testing to find the maximum capacity of an application.

Downward scalability testing

The downward scalability testing is used when the load testing is not passed, then start **decreasing the no. of users in a particular interval** until the goal is achieved. So that it is easy to identify the bottleneck (bug).

Security Testing

Security testing is used to detect the security flaws of the software application. The testing is done via investigating system architecture and the mindset of an attacker. Test cases are conducted by finding areas of code where an attack is most likely to happen.

Portability Testing

The portability testing of the software is used to verify whether the system can run on different operating systems without occurring any bug. This test also tests the working of software when there is a same operating system but different hardware.

Reliability Testing

Reliability test assumes that whether the software system is running without fail under specified conditions or not. The system must be run for a specific time and number of processes. If the system is failed under these specified conditions, reliability test will be failed.

Efficiency Testing

Efficiency test examines the number of resources needed to develop a software system, and how many of these were used. It also includes the test of these three points.

- Customer's requirements must be satisfied by the software system.
- A software system should achieve customer specifications.
- Enough efforts should be made to develop a software system.

Advantages of Non-functional testing

- It provides a higher level of security. Security is a fundamental feature due to which system is protected from cyber-attacks.
- It ensures the loading capability of the system so that any number of users can use it simultaneously.
- It improves the performance of the system.
- Test cases are never changed so do not need to write them more than once.
- Overall time consumption is less as compared to other testing processes.

Disadvantages of Non-Functional Testing

- Every time the software is updated, non-functional tests are performed again.
- Due to software updates, people have to pay to re-examine the software; thus software becomes very expensive.

13. Other types of testing:

(a)Regression Testing-This testing is done to make sure that new code changesshould not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

(b) Smoke Testing- This testing is done when we receive new build to check whether the built is stable for testing or not. Smoke testing mainly focus on the corefunctionality of the application.

(c) Sanity Testing- This testing is done ***after Smoke Testing*** on the new build to verify that all bug have been fixed and there is no other bug originated with new changes in the built. It is asubset of a regression.

(d) Recovery Testing- It is type of software testing which verify software

ability torecover from failure like hardware or software, network failure or power cut, etc.

Example: Suppose we are using the browser, let say **Google Chrome**, and the power goes off. When we switch on the system again and re-open Google Chrome, we get a message window that displays whether we want to start a new session or restore the previous session.

So, in this situation, while we are restarting the system while a browser has a definite number of sessions and check if the browser can recover all of them or not.

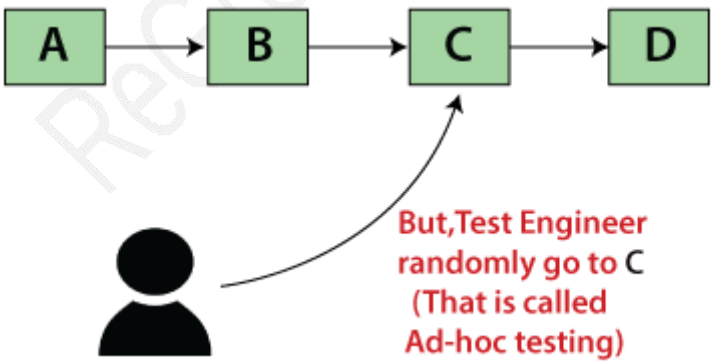
(e) Alpha testing- Alpha testing is the first end-to-end testing of a product to ensureit meets the business requirements and functions correctly. It is typically performed ***by internal employees*** and conducted in a lab/stage environment, before it is released to actual end-user.

(f) Beta Testing- It is a type of acceptance testing performed ***by the real user in the real environment*** to verify all the functionality of the software before releasingthe software to final production.

(g) Ad Hoc Testing- It is a type of software testing performed without proper planning and documentation. This testing ***is done after formal testing*** is performed onthe software.

Example:

In Adhoc testing, we don't follow the requirement because we randomly check the software. Our need is A->B->C->D, but while performing Adhoc testing, the test engineer directly goes to the C and test the application as we can see in the below image:



(h) Monkey testing- Monkey testing is a type of software testing in which a software or application is tested using ***random inputs*** with the sole purpose of tryingand breaking the system. There are no rules in this type of testing. It completely works on the tester's mood or gut feeling and

experience.

(i) Agile testing- It is type of software testing practice that follows the principle ofagile software development.

(j) Retesting testing- Testing of only that particular bug again after fixing it.

(k) Exploratory Testing- Exploratory testing allows you to think outside the box and come up with use cases that might not be covered in a test case. For example, youmight perform one test and then ask yourself, What if I tried this? What if I didn't dothat?

For example, to test any software or the application, first, we will perform unit, integration, and system testing.

So if we want to understand any application first, we will perform unit or component testing, suppose the application is having a login page having many elements, and we will understand each part and doing the component testing, but actually, we are doing the exploratory testing because we are exploring the application.

Suppose we have many modules in the application, and we are trying to do some integration scenarios.

Indirectly we are just doing exploratory testing while performing the integration testing.

And, even if we are performing system testing, indirectly, we are performing exploratory testing because here we are also understanding and exploring the application.

14. Bug Life Cycle:

Set of states that bug goes through in its entire life.

New-When a bug is found while testing, its status is assigned as new.

Assigned- Once the bug is posted by the tester, the lead of the tester approvesthe bug and assigns the bug to the developer team.

Open- The developer starts analyzing and works on the defect fix, if required. Ifdeveloper feels that the defect is not appropriate then it may choose any of the state (deferred, rejected, duplicate, not a bug) along with the reason.

Fixed- When developer fix the bug, he mark the status of defect as fixed andassign to the tester for re-testing.

Re-testing- Tester do the re-testing and verify the defect again.

Re-open- If any issue occurs again in that defect then it will again assign to thedeveloper and mark defect status as re-open.

Verified- If after fixing the defect, testing is done and tester feels that defect hasbeen fixed then defect status is marked as verified.

Closed- When defect does not exist any longer then tester changes the defectstatus as closed.

Duplicate: If the defect is repeated twice or the defect corresponds to the sameconcept of the bug, the status is changed to "duplicate."

Rejected: If the developer feels the defect is not a genuine defect then it changesthe defect to "rejected."

Deferred: If the present bug is not of a prime priority and if it is expected to getfixed in the next release, then status "Deferred" is assigned to such bugs

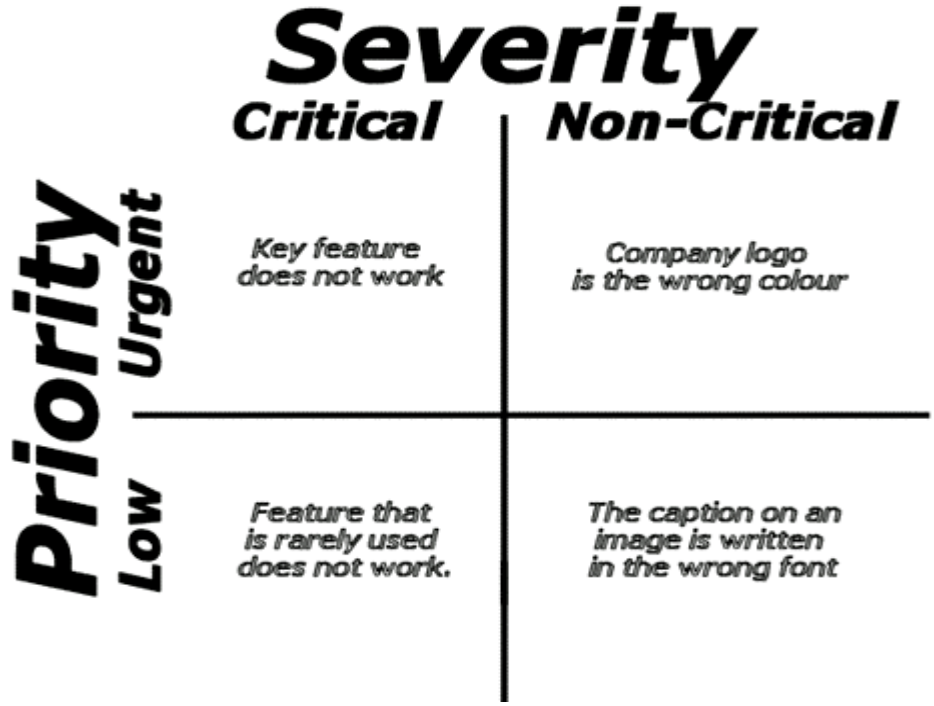
Not a bug: If it does not affect the functionality of the application then the statusassigned to a bug is "Not a bug".

15. Severity and Priority in testing:

Severity- Tells us about the defect impact on the application (how bad or dangerous the defect is).It is given by the QA tester. It can be a blocker, critical,major, and minor for the bug.

Blocker: if the severity of a bug is a blocker, which means we cannot proceed tothe next module.

Priority- Tell us about how soon the bug should be fixed. It can be urgent, high,medium, and low. It is given by the test lead or the project manager.



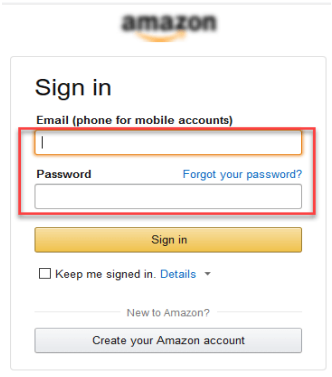
16. Difference between test scenario and test cases:

Test Scenario- It gives us the idea of what we have to test.e.g., verify the login functionality of Gmail account.

Example: Test Scenario for e-Commerce Application

For an e-Commerce Application, a few test scenarios would be

Test Scenario 1: Check the Login Functionality



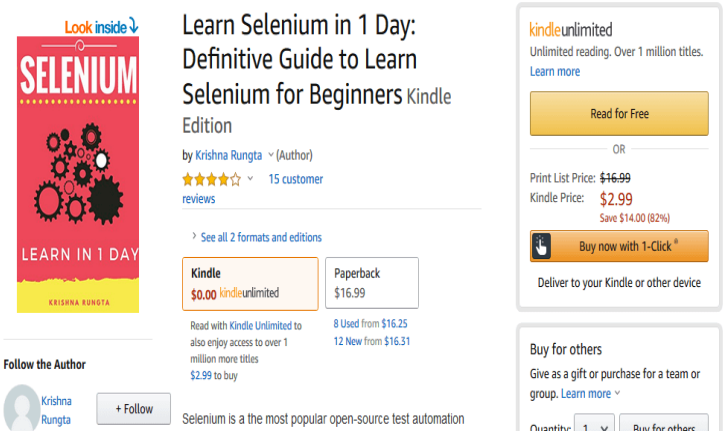
In order to help you understand the difference Test Scenario and Test Cases, specific test cases for this Test Scenario would be

1. Check system behaviour when valid email id and password is entered.

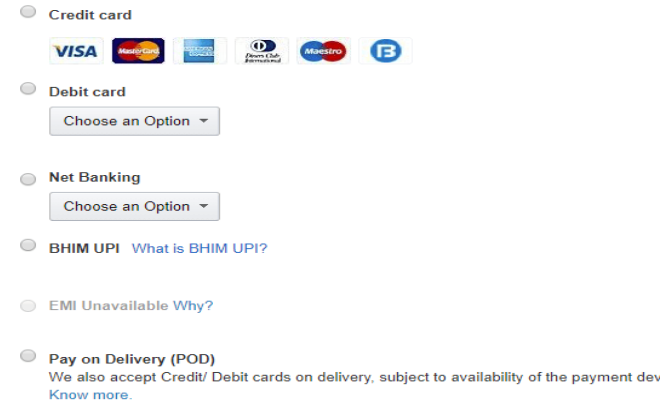
2. Check system behaviour when *invalid* email id and *valid* password is entered.
3. Check system behaviour when *valid* email id and *invalid* password is entered.
4. Check system behaviour when *invalid* email id and *invalid* password is entered.
5. Check system behaviour when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behaviour when valid/invalid phone number and password is entered.
8. Check system behaviour when “Keep me signed” is checked

Test Scenario 2: Check the Search Functionality

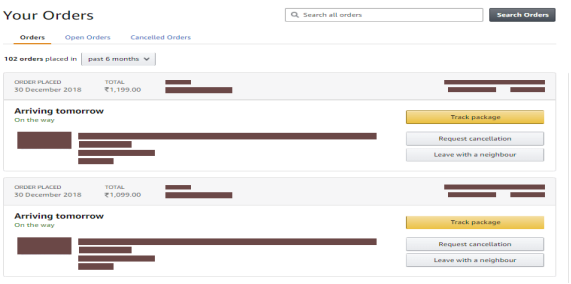
Test Scenario 3: Check the Product Description Page



Test Scenario 4: Check the Payments Functionality



Test Scenario 5: Check the Order History

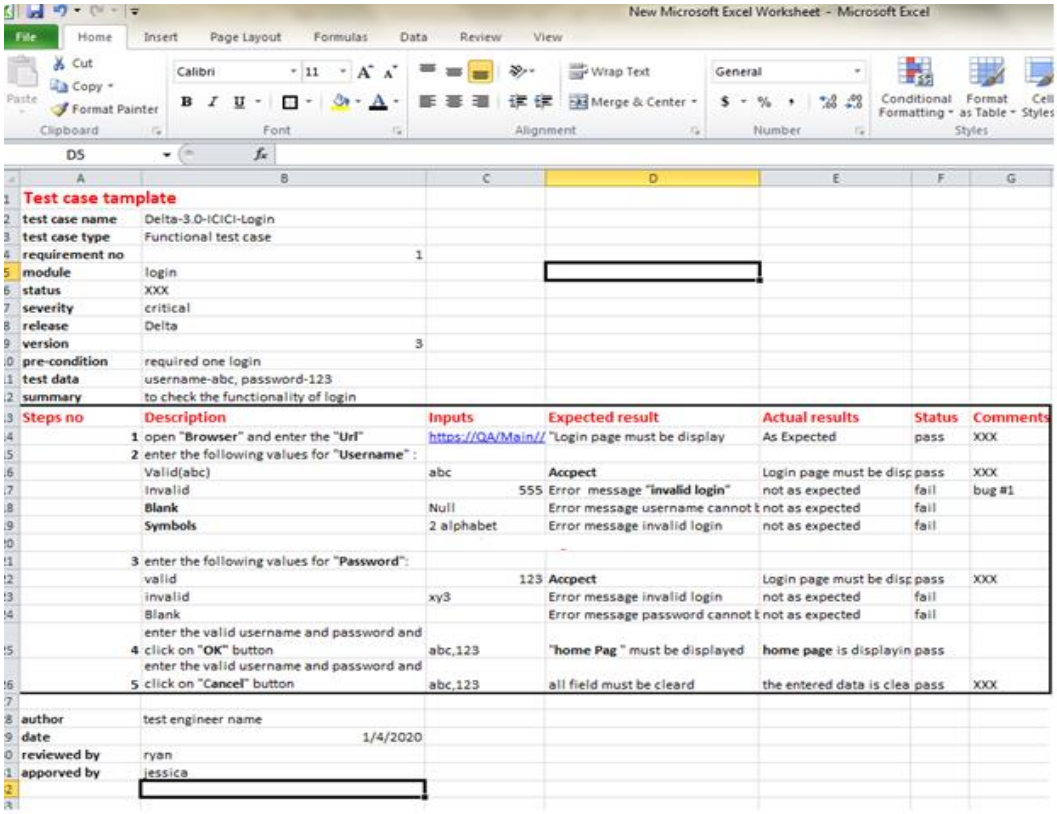


Test Case- It gives us the idea on how to test it.

e.g., enter valid username, valid password, enter invalid username, enter invalidpassword.

Test Case Parameters

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Actual Result
- Status
- Bug id
- Bug Status



New Microsoft Excel Worksheet - Microsoft Excel

Test case tamplate						
test case name	Delta-3.0-ICICI-Login					
test case type	Functional test case					
requirement no	1					
module	login					
status	XXX					
severity	critical					
release	Delta					
version	3					
pre-condition	required one login					
test data	username-abc, password-123					
summary	to check the functionality of login					
Steps no	**Description**	**Inputs**	**Expected result**	**Actual results**	**Status**	**Comments**
1	open "**Browser**" and enter the "**Url**"	https://QA/Main//	"Login page must be display	As Expected	pass	XXX
2	enter the following values for "**Username**" :					
	Valid(abc)	abc	**Accpect**	Login page must be disp	pass	XXX
	Invalid	555	Error message "**invalid login**"	not as expected	fail	bug #1
	Blank	Null	Error message username cannot b	not as expected	fail	
	Symbols	2 alphabet	Error message invalid login	not as expected	fail	
3	enter the following values for "**Password**":					
	valid	123	**Accpect**	Login page must be disp	pass	XXX
	invalid	xy3	Error message invalid login	not as expected	fail	
	Blank		Error message password cannot b	not as expected	fail	
4	enter the valid username and password and click on "**OK**" button	abc,123	"**home Pag** " must be displayed	**home page** is displayin	pass	
5	enter the valid username and password and click on "**Cancel**" button	abc,123	all field must be cleard	the entered data is clea	pass	XXX
author	test engineer name					
date	1/4/2020					
reviewed by	ryan					
apporved by	jessica					

17. Boundary Value Analysis:

Boundary Value Analysis (BVA) is a Black-Box testing technique used to check the errors at the boundaries of an input domain. The name comes from the Boundary,which means the limits of an area.

Ex., Imagine, there is a function that accepts a number between 18 to 30, where 18 isthe minimum and 30 is the maximum value of valid partition, the other values of thispartition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition

Consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and

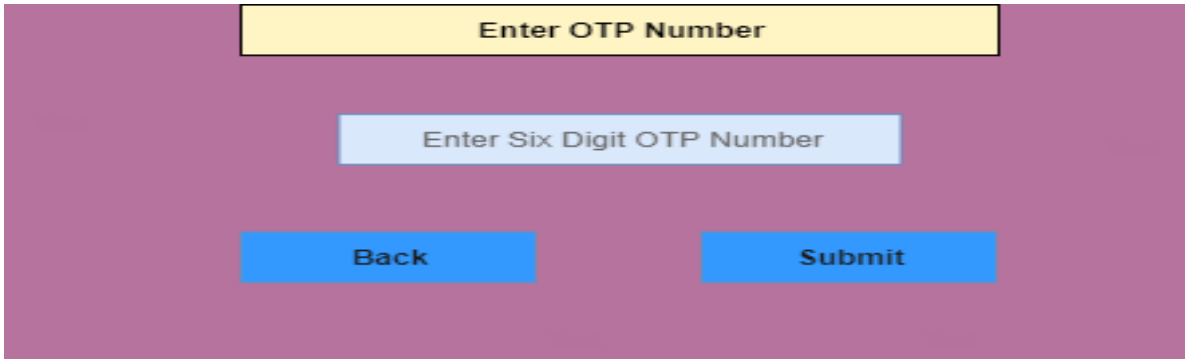
more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both validand invalid partitions to capture the behavior of the system on different input conditions.

18. Equivalence Partitioning Technique

Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence

partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other.

OTP Number = 6 digits



INVALID	INVALID	VALID	VALID
1 Test case	2 Test case	3 Test case	
DIGITS >=7	DIGITS<=5	DIGITS = 6	DIGITS = 6
93847262	9845	456234	451483

19. Requirement Traceability Matrix (RTM):

Is a document that maps and traces user requirement with test cases. It captures allrequirements proposed by the client and requirement traceability in a single document. 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.

Which Parameters to include in Requirement Traceability Matrix?

- Requirement ID
- Requirement Type and Description
- Test Cases with Status

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

20. Diff. between error, defect, bug, failure:

A mistake in coding is called **Error**, error found by tester is called **Defect**, defectaccepted by development team then it is called **Bug**, build does not meet the requirements then it Is **Failure."**

21. Entry and Exit criteria

- **Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.
- **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded.

22. Diff. between Authentication and Authorization

Authentication	Authorization
Authentication is the process of identifying a user to provide access to a system.	Authorization is the process of giving permission to access the resources.
In this, the user or client and server are verified.	In this, it is verified that if the user is allowed through the defined policies

	and rules.
It is usually performed before the authorization.	It is usually done once the user is successfully authenticated.
It requires the login details of the user, such as user name & password, etc.	It requires the user's privilege or security level.

23. Defect Rejection Ratio:

DRR= (no of defects rejected / total no. of defect raise)*100

24. Defect Leakage Ratio:

DLL= (no of defect missed/ total defect of software)*100

25. Defect Density:

Defect density= defect count/ size of the release

Size of release can be measured in terms of line of code(Loc)

Example-

Module 1= 10 bug

Module 2= 20 bug

Module 3= 30 bug

Total bug= 60

Loc in each module

Module 1= 1000 loc

Module 2= 1500 loc

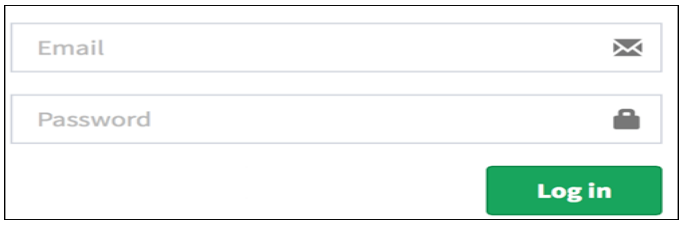
Module 3= 500 loc, Total loc=3000

26. What is Decision Table Testing

Decision table testing is a software testing technique used to test system behaviour for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behaviour (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage.

Example: How to make Decision Base Table for Login Screen

Let's create a decision table for a login screen.



The condition is simple if the user provides the correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
Password (T/F)	F	F	T	T
Output (E/H)	E	E	E	H

- **T** – Correct username/password
- **F** – Wrong username/password
- **E** – Error message is displayed
- **H** – Home screen is displayed

- **Case 1** – Username and password both were wrong. The user is shown an error message.
- **Case 2** – Username was correct, but the password was wrong. The user is shown an error message.
- **Case 3** – Username was wrong, but the password was correct. The user is shown an error message.
- **Case 4** – Username and password both were correct, and the user navigated to the homepage

27. What is State Transition Testing

State Transition Testing is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test (AUT). State transition testing helps to analyse

behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behaviour.

Four Parts of State Transition Diagram

There are 4 main components of the State Transition Model as below

1) States that the software might get

2) Transition from one state to another

3) Events that origin a transition like closing a file or withdrawing money

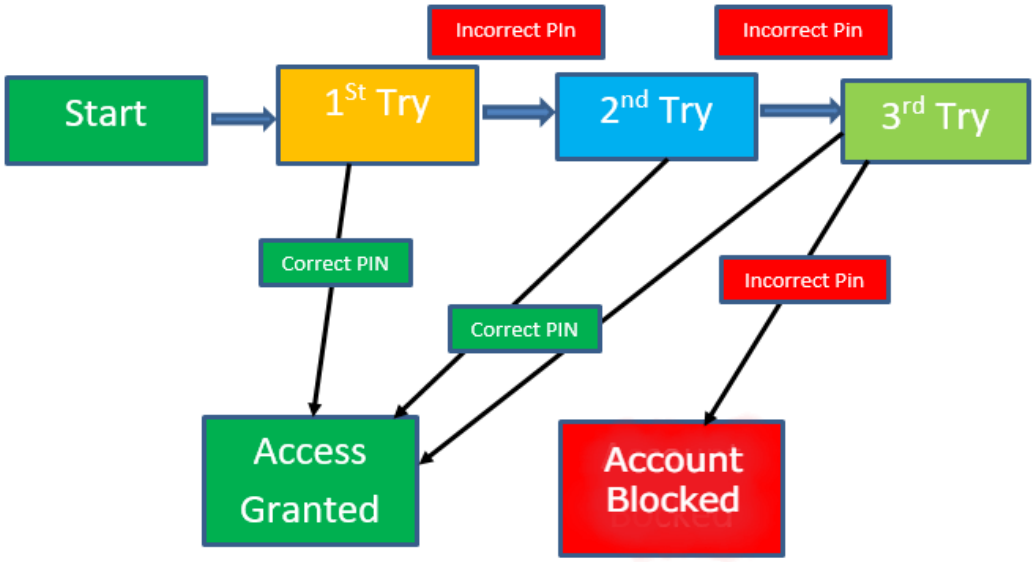
4) Actions that result from a transition (an error message or being given the cash.)

Example:

Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

State transition diagram



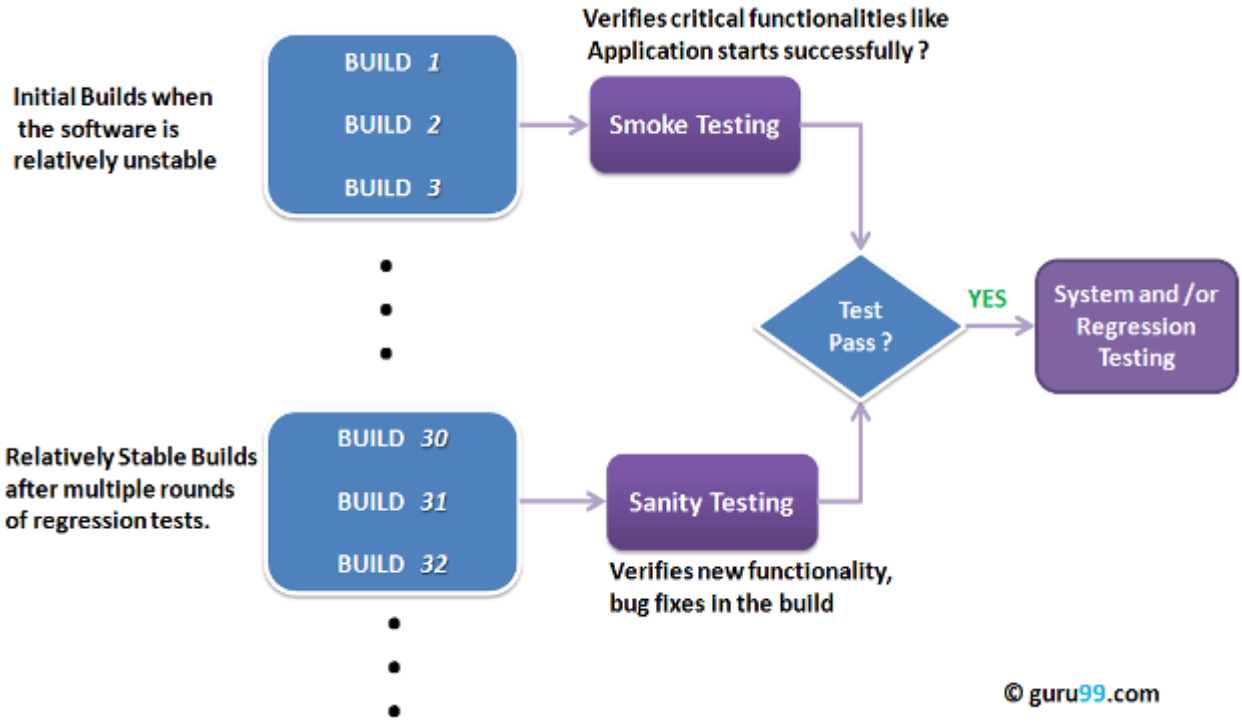
In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

28. Difference between Load testing and Stress testing

Load testing	Stress testing
Load testing is used to find the performance of the application by testing the database, networks, and website servers.	Stress testing is used to find the stability and response time of the given system.
This type of testing reproduced the load on any application and software.	It is used to figure out the robustness and stability of the application.
Load testing is used to test web-based and client-server types of application.	Stress testing tests suddenly increased traffic of the application.
For example: if we have one scenario where the load is 100 users using the application at a 2.5\sec of goal time. And, the desired load is 100 users. This scenario got passed because the	For example: if we took the same scenario where the actual load of 100 users which are using the application at a 2.5\sec of goal time. When the desired load got increased by 200

desired load is equal to the load, which satisfies the load testing condition.	users, and it will become 300. So now, 300 users using the application at 2.5\sec of goal time. It will pass because the desired load is greater than the load according to the stress testing condition.

29. Smoke Testing vs Sanity Testing



S.No.	Comparison Basis	Smoke Testing	Sanity Testing
1	Test coverage	It is a broad approach to testing where all parts of the application are tested.	It is a narrow approach to testing where specific parts of the application are tested.
2	Measures	It measures the stability of the system by performing rigorous testing.	It measures the rationality of the system by performing rigorous testing.
3	Technique	Smoke testing can be either	Sanity testing can be done

		manual or automated.	without test cases or scripts.
4	Executed by	It is performed by both testers and developers.	It is performed by only testers.
5	Purpose	Testing is done without getting into deep but whenever needed tester has to go into deep.	Sanity testing does not need to go into deep of the application.
6	Subset	It is considered as a subset of acceptance testing.	It is considered as a subset of regression testing.

30. System Testing VS. Acceptance Testing

S.no.	Comparison basis	System Testing	Acceptance Testing
1	**Definition**	System testing is performed to test end to end functionality of the software.	Acceptance testing is performed to test whether the software is conforming specified requirements and user requirements or not.
2	**Executed by**	Only developers and testers can perform System testing.	It can be performed by testers, stakeholders and costumers.
3	**Part of**	It can be both non-functional and functional testing.	It can be only functional testing.
4	**Analysis**	In System testing, we test the performance of the whole system.	In Acceptance testing, we test whether the system is conforming requirements or not.
5	**Order of execution**	It is performed before the Acceptance testing.	It is performed after the System testing.

31. Differences between the Alpha testing and Beta testing

sr. No.	Alpha Testing	Beta Testing
1.	Alpha testing performed by a team of highly skilled testers who are usually the internal employee of the organization.	Beta testing performed by clients or end-users in a real-time environment, who is not an employee of the organization.
2.	Alpha testing performed at the developer's site; it always needs a testing environment or lab environment.	Beta testing doesn't need any lab environment or the testing environment; it is performed at a client's location or end-user of the product.
3.	Reliability or security testing not performed in-depth in alpha testing.	Reliability, security, and robustness checked during beta testing.
4.	Alpha testing involves both white box and black-box techniques.	Beta testing uses only black-box testing.
5.	Long execution cycles maybe require for alpha testing.	Only a few weeks are required for the execution of beta testing.
6.	Critical issues or fixes can be identified by developers immediately in alpha testing.	Most of the issues or feedback is collecting from the beta testing will be implemented for the future versions of the product.
7.	Alpha testing performed before the launch of the product into the market.	At the time of software product marketing.
8.	Alpha testing focuses on the product's quality before going to beta testing.	Beta testing concentrates on the quality of the product, but gathers users input on the product and ensures that the product is ready for real-time users.
9.	Alpha testing performed nearly the end of the software development.	Beta testing is a final test before shipping a product to the customers.
10.	Alpha testing is conducting in the presence of developers and the absence of end-users.	Beta testing reversed of alpha testing.

32. What is Globalization testing?

It is another type of software testing used to test the software developed for multiple languages, is called **globalization testing**, and improving the application or software for various languages is known as **globalization**.

Globalization testing ensures that the application will support multiple languages and multiple features because, in current scenarios, we can see the enhancement in several technologies as the applications are planned to be used globally.

For example, www.google.com supports many languages, and people from different countries can access it; therefore, it is a globalized product.

33. What is Localization testing?

It is nothing but a format of **software testing**. We test the particular application based on the country, region, etc. As we know, the Localized product only supports the precise kind of languages that are usable only in a specific region.

It is also known as **L10N** testing, and here **10** is the number between **L** and **N** in the **L**localizatio**n** word.

For example, QQ.com supports only the Chinese language, which can be accessed only by a few countries.

33. What is Test Plan?

The **test plan** is a base of **software testing**. It is a detailed document, which includes several testing attributes such as **test objectives, scope, test schedule, template, required resources (human resources, software, and hardware), test estimation and test deliverables, risk, mitigation plan, defect tracking, entry and exit criteria, test environment**, etc., which defines software testing areas and activities.

The test plans play a major role in testing and help us deliver a quality product.

It is derived with the help of Use Case documents, SRS (Software Requirement Specification), and Product Description.

A Test Plan is a dynamic document that can be updated frequently when new requirements or modifications have occurred.

34. What is a Test Strategy?

The test strategy is a high-level document used to validate the test levels to be executed for the product. And it also describes what kind of technique has to be used and which module will be tested.

It contains various components like **documentation formats, objectives, test processes, scope, customer communication strategy,** etc.

The Test Strategy's main purpose is to deliver a systematic approach to the software testing process to ensure **reliability, quality, traceability, and better planning**.

While the Test Strategy can be derived with the help of the BRS (Business Requirement Specification) document.

It is a static document, which implies that it cannot be changed or modified.

35. What is Positive Testing?

It is used to check whether our application works as expected or not. And if an error is detected at the time of positive testing, the test is considered as fail. A positive testing is a technique whenever a test engineer writes the test cases for a set of respective outputs.

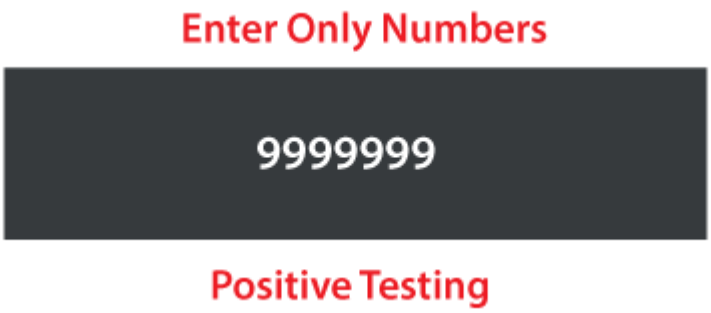
In positive testing, the test engineer will always check for only a good set of values.

In other words, we can say that **positive testing** is a process where the system or an application is tested against the valid input data.

And the primary purpose of performing the positive testing is to validate whether the software does what it is supposed to do.

In simple terms, we can say that positive testing is implemented by providing a **positive point of view.**

For example, **Numbers like 9999999.**



In the following example, we are trying to understand the working of positive testing.

- Suppose, we have one test scenario, where we want to test an application that includes a simple text box.
- To enter **Phone Number** according to the business needs it accepts only numerical values.
- Hence, in positive testing, we will only give the positive numerical values in order to test whether it is working as per the requirement or not.

36. What is Negative Testing?

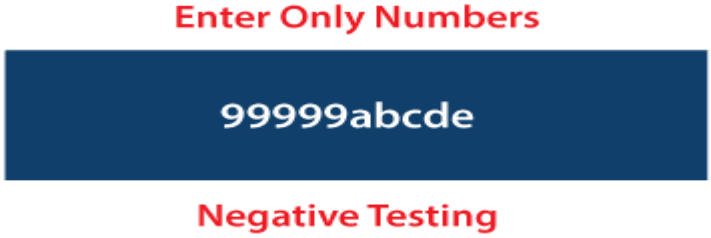
It is implemented to check how the application can gracefully handle invalid input or unpredicted user performance.

The fundamental purpose of executing the negative testing is to ensure the application's stability against the effects of different variations of improper validation data set.

Negative testing is also known as **error path testing or failure**. And it helps us to identify more bugs and enhance the quality of the software application under test.

Once the positive testing is complete, then only we can execute the negative testing, which helps to identify more bugs and enhance the quality of the software application under test.

We can say that the negative testing is executing by keeping the **negative point of view** in simple terms. For example, **99999abcde**



Let see one example which will help us to understand the negative testing in an effective way.

- Suppose we have one sample form where the **Mobile Number** field can only accept the numbers' value and does not accept the unique characters and alphabets values.
- However, let's entered the unique characters and alphabets values on **the Mobile number** field to test that it accepts the individual characters and alphabets values or not.
- We expect that the textbox field will not accept invalid values and show an error message for the incorrect entry.

37. What is Bug release?

- Bug release is software or an application is handed over to the testing team knowing that the defect is present in a release. During this the severity and priority of bug is low, as bug can be removed before the final handover.

38. What is defect cascading?

Defect cascading is when one defect leads to the discovery of other defects. This can happen for a variety of reasons, but often it occurs because the original defect was not fixed properly. When this happens, it can have a cascading effect on the entire development process, as each new defect that is discovered can lead to more delays and additional costs.

39. How can you prevent defect cascading from happening in your own software testing process?

There are a few things you can do to prevent defects cascading in your own software testing process:

Fix the original defect properly

The best way to prevent defect cascading is to fix the original defect properly. This can be done by ensuring that all defects are fixed before the product is released. To do so, you'll need to have a robust quality assurance process (and, ideally, a dedicated QA team) in place that can catch defects before they become part of the product.

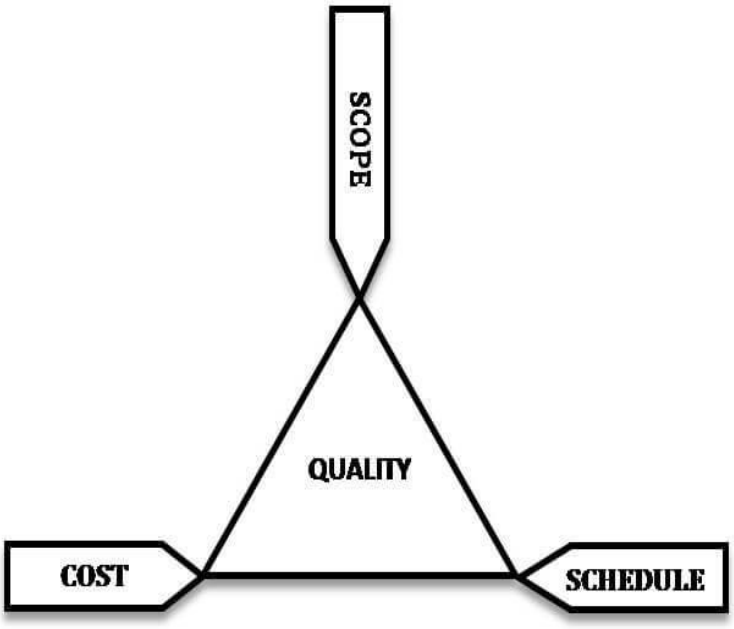
Communicate effectively

Another important way to prevent defect cascading is to ensure that there is good communication between the different teams involved in the development process. This includes the software development team, the quality assurance team, and the customer support team.

40. Software Testing Metrics

Software Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.



Software testing metrics or software test measurement is the quantitative indication of extent, capacity, dimension, amount or size of some attribute of a process or product.

Why Test Metrics are Important?

"We cannot improve what we cannot measure" and Test Metrics helps us to do exactly the same.

- Take decision for next phase of activities
- Evidence of the claim or prediction
- Understand the type of improvement required
- Take decision or process or technology change

Test Metrics Life Cycle

Different stages of Metrics life cycle	Steps during each stage
• Analysis	• Identification of the Metrics • Define the identified QA Metrics
• Communicate	• Explain the need for metric to stakeholder and testing team • Educate the testing team about the data points to need to be captured for processing the metric
• Evaluation	• Capture and verify the data • Calculating the metrics value using the data captured
• Report	• Develop the report with an effective conclusion • Distribute the report to the stakeholder and respective representative • Take feedback from stakeholder

Example of Test Metric

To understand how to calculate the test metrics, we will see an example of a percentage test case executed.

To obtain the execution status of the test cases in percentage, we use the formula.

Percentage test cases executed= (No of test cases executed/ Total no of test cases written) X 100
Likewise, you can calculate for other parameters like **test cases not executed, test cases passed, test cases failed, test cases blocked, etc.**

41. What is Defect Triage?

Defect triage is a process in software testing where it defines the order of defects which will be resolved according to the priority of severity or risks etc. It is also called Bug Triage. It is based on the severity and priority of the defects in software.

Severity means the degree of impact or amount of impact of the defect on the software or part of the application which is tested. Priority indicates the correct order in which the detected defects should be fixed or resolved.

So basically, Defect Triage helps the software testing team if the available resources are very few to fix the bugs in software.

'Triage' is a **French** word which means **'sorting'**. So Defect Triage gives the degree of risk to each bug in software testing. If the number of bugs is too much than the number of testers, then this process automatically tries to fix the bugs as much as possible based on the degree of risks.