What is software testing?

Software testing Is a process to evaluate/calculate the functionality of a software application with an intent/aim to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce a quality product.

Testing is a process of verification and validation.in verification we check whether the features requested by the client is present or not and whether those features are present in proper place or not.

In validation we check whether those features are functionally working fine or not.

What is manual testing?

Manual testing can be defined as the verifying the functionality of an application manually without using any automated tool or script.

In manual testing the tester takes over the role of an end user and tests the software to identify any unexpected behavior or bug.

What is automation testing?

Verifying the functionality of an application automatically by writing automation codes through some programing language like java, python, java script.

We need a platform/environment for writing this automation code like eclipse IDE, IntelliJ IDE, net Beans IDE.

We need some automation tool to execution the testing like selenium, QTP, remorex text complete.

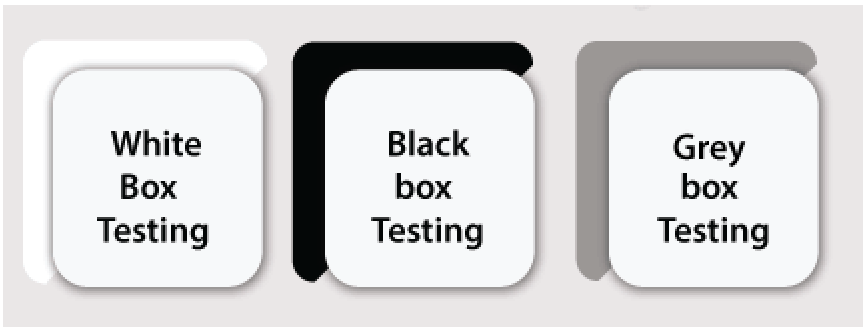
Why we do software testing?

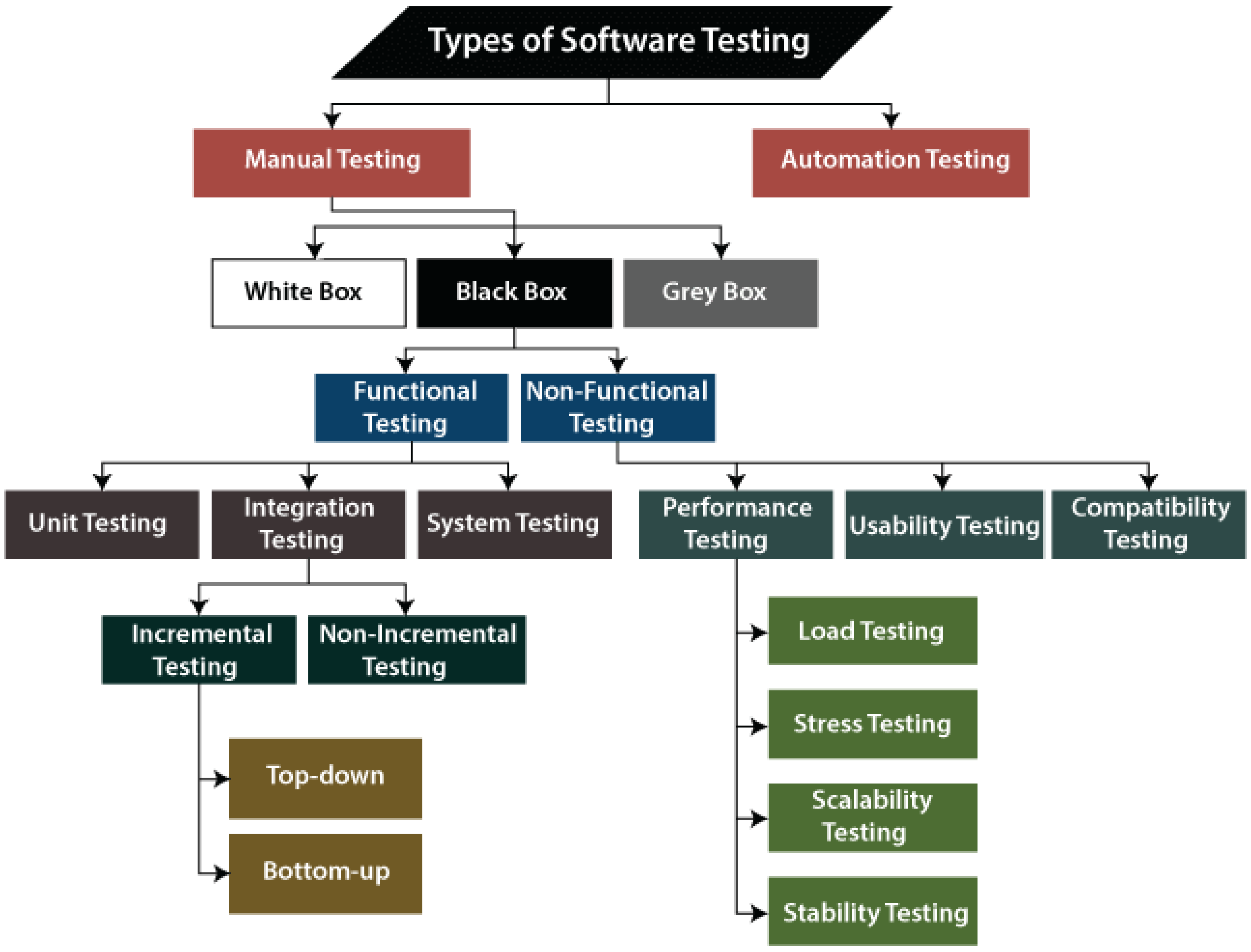
Every software is developed to support the business of the customer. If there are any defect, it will affect the business of the customer, because a bad name will spread in the market and number of end users will reduce which in turn will affect the business of the customer. So, in order to deliver a quality product to the customer, we should perform software testing.

Types of software testing---

In software testing, manual testing can be further classified into three different types of testing. These are—

1. White box testing
2. Black box testing
3. Gray box testing





I. White box testing—

White box testing is also known as glass box testing, structural testing, clear box testing, open box testing and transparent box testing.

Developer does white box testing where they test every line of code of the program. In white box testing programing skills are required to design test cases.

The developers perform the white box testing and then send the application or software to the testing team. The white box testing contains various test, which are-

- ➢Path testing
- ➢Loop testing
- ➢Condition testing
- ➢Testing based on the memory perspective
- ➢Test performance of the program

II. Black box testing—

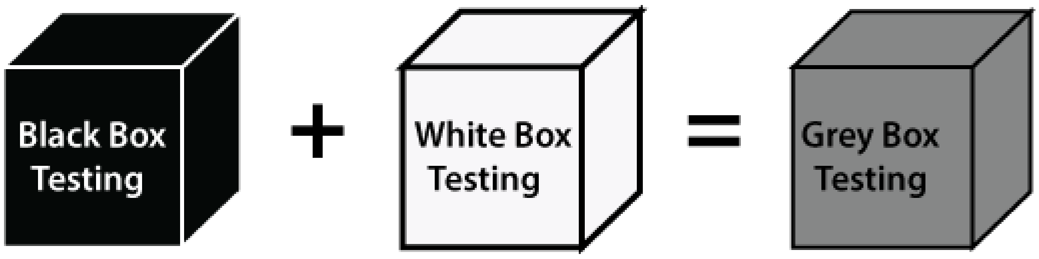
Another type of manual testing is black box testing. Black box testing is also known as close box testing, functional testing or behavioral testing.

Black box testing is a software testing method test by the test engineer, in which the functionality of the software applications is tested without having knowledge of internal code structure, implementation details and internal paths.

Black Box Testing

III. Gray box testing---

Another part of manual testing is gray box testing. Gray box testing is a software testing technique which is a combination of black box testing technique and white box testing technique.



In black box testing technique, tester is unknown to the internal structure of the item being tested and in white box testing the internal structure is known to the tester. The internal structure is partially known in grey box testing.

Why grey box testing---

Gray box testing is performed for following reasons: -

- It provides combine benefits of both black box testing and white box testing both.
- It combines the input of developers as well as testers and improves overall product quality.
- It provides enough free time to developers to fix defects.
- To test from the user point of view rather than a designer point of view.

DIFFERENT BETWEEN WHITE BOX TESTING AND BLACK BOX TESTING-

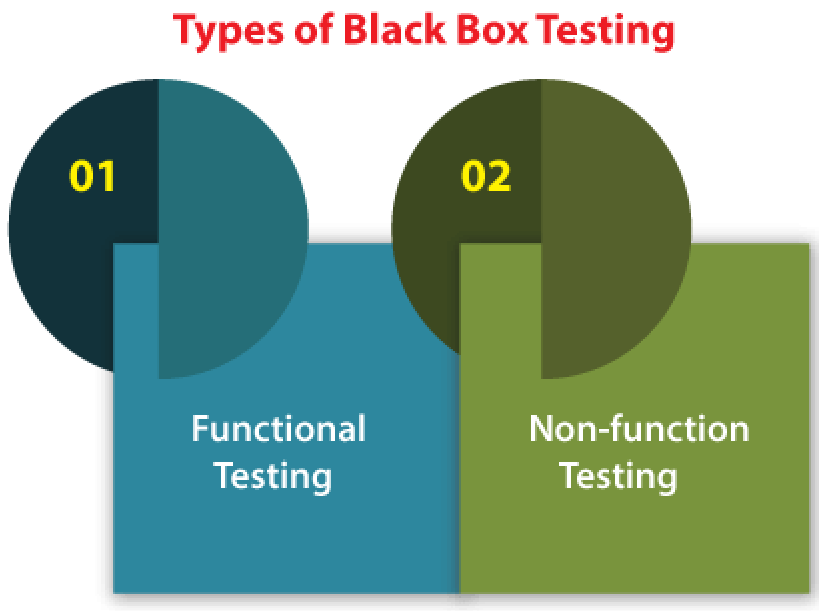
White box testing	**Black box testing**
❖ White box testing is mostly done by the software developers.	❖ Black box testing is mostly done by the software tester.
❖ It is a testing approach in which internal structure is known to the tester.	❖ It is a testing approach which is used to test the software without the knowledge of the internal structure of
❖ Programing knowledge is required to perform white box testing.	❖ Programming knowledge is not needed to perform black box testing

❖ It is the logic testing of the software. ❖ It is generally applicable to the lower level of software testing. ❖ It is the most time consuming.	❖ It is the behavior testing of the software. ❖ It is applicable to the higher levels of testing of software. ❖ It is least time consuming.

Types of black box testing---

Black box testing further categorized into two parts. these are—

1)Functional testing
2)Nonfunctional testing



Functional testing also classified into 11 types. These are—

I. Functionality testing
II. Integration testing
III. System testing
IV. Acceptance testing
V. Smoke testing
VI. Sanity testing
VII. Adhoc testing
VIII. Exploratory testing
IX. Regression testing
X. Compatibility testing
XI. Globalization testing

Nonfunctional testing also classified into 5 types. These are----

I. Performance testing
II. Usability testing
III. Reliability testing
IV. Recovery testing
V. Accessibility testing

i. Functionality testing---

It is a type of software testing which is used to verify the functionality of the software application, where the function is working according to the requirement specification.

In functional testing, each function tested by giving the value, determining the output, and verifying the actual output with expected value.

Functionality testing performed as a black box testing which is presented to confirm that the functional of an application or system behaves as we are expecting.

Approach to do functional testing---

Always testing starts by giving valid data. If the component is working fine for the valid data, then test the same component by giving multiple invalid data.

Note—while testing a component by giving valid data, other component should be blank.

While testing a component by giving invalid data, other component should be valid.

Test scenario/test data are generally derived by referring requirement document(srs).

Types of functionality testing—

There are two type of functionality testing. These are—

- +ve functionality testing.
- -ve functionality testing.

  In +ve functionality testing we are testing the component of an application by giving valid or +ve data (as per the requirement of the customer In -ve functionality testing we are giving invalid data as per the against the requirement of the customer.

Note- while doing functionality testing, we should not do over testing and under testing rather we should do only optimize testing.

Over testing: --testing the component of an application by giving data that does not make any sense (junk data). If we do over testing, we are going to waste our testing time.

Under testing: ---testing the component of an application by giving insufficient scenario. If we do under testing, we might miss to catch some of the defect present in the application.

Optimized testing: --testing the component of an application by giving data which really make sense.

ii. Integration testing/data flow testing---

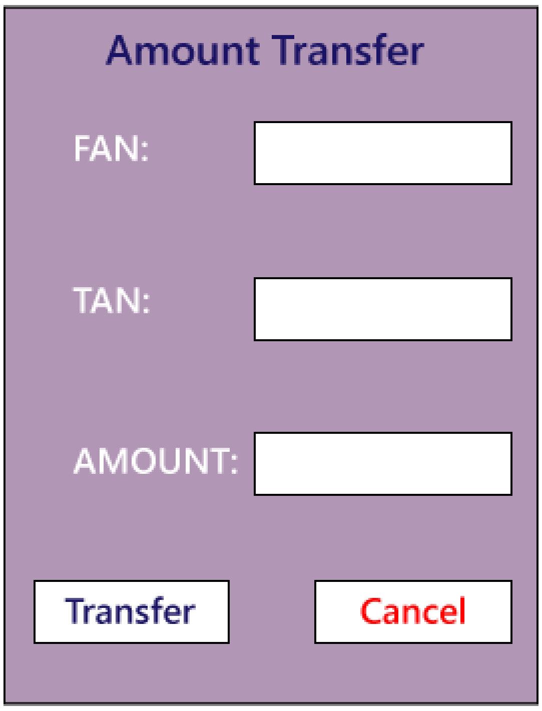
Integration testing is the type of testing in which we test the data flow between any two dependent modules of an application. Before we perform integration testing, functionality testing scenario should be fine. In order to perform integration testing, minimum two features are required and both features should be functionally working fine.

Can data flow happen between two modules of the same application? Yes

Can data flow happen between two modules of two different application? Yes

Can data flow happen between two modules of two different application belonging to the same project? Yes

Example of integration testing—

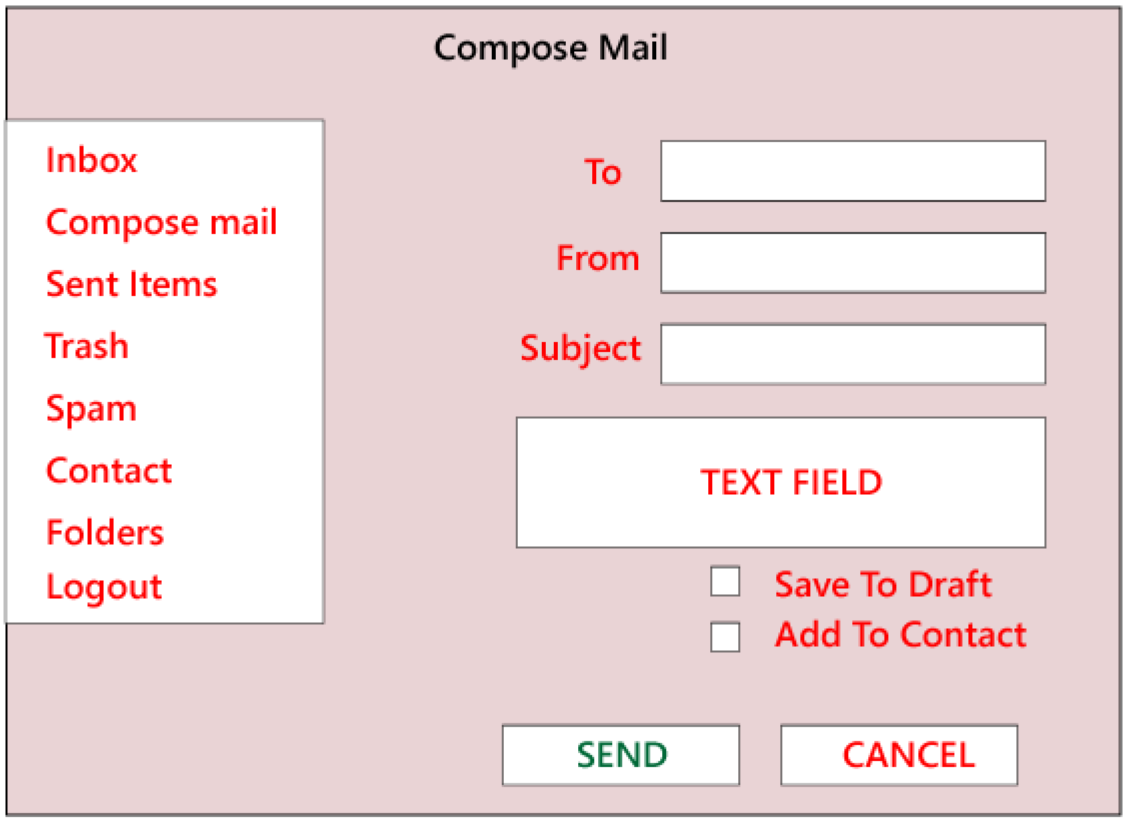


- First, we will login as user P to transfer amount and send rupees 200, the confirmation message should be displayed on the screen as **amount transfer successfully.** now logout as p and login as user Q and go to amount balance page and check for balance in that account=present balance+received balance. there for the integration test is successful.
- Also, we check if the amount of balance has reduced by Rs 200 in p user account.
- 3Click on the transaction, P and Q, the message should be displayed regarding the date and time of the amount transfer.

Example 2---

Let us assume that we have a Gmail application where we perform the integration testing.

First, we do functional testing on the login page, which include the various component such as username, password, cancel and submit button. Then only we can perform integration testing.



Scenario 1---

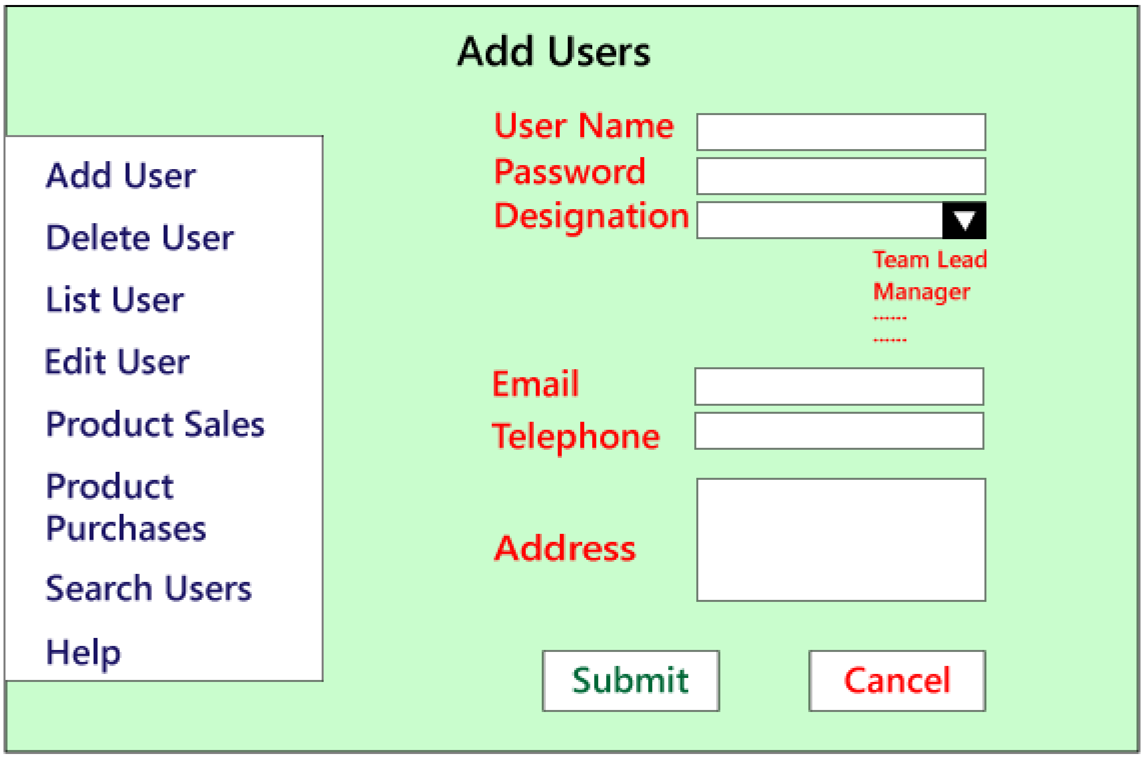
- First, we login as P users and click on the compose mail and performing the functional testing for the specific component.
- Now we click on the send and also check for save drafts.
- After that we send a mail to Q and verify in the send items folder of P to check if the send mail is there.
- Now we logout as P and login as Q and move to the inbox and verify that if the mail has reached.

Scenario 2---

We also perform the integration testing on spam folder. If the particular contact has been marked as spam, then any mail sent by that user should go to the spam folder and not in the inbox.

As we can see in the below image, we perform the functional testing for all the text fields and every feature.

Then we will perform integration testing for the related functions. We first test the add user, list of users, delete user, edit user, and then search user.



<u>Approach to do integration testing—</u>

I. Go through(understand) the requirement of the client.
II. Understand how the features functionally works.
III. Understand how the features are inter related to each other.
IV. Derive the integration/data flow scenarios.

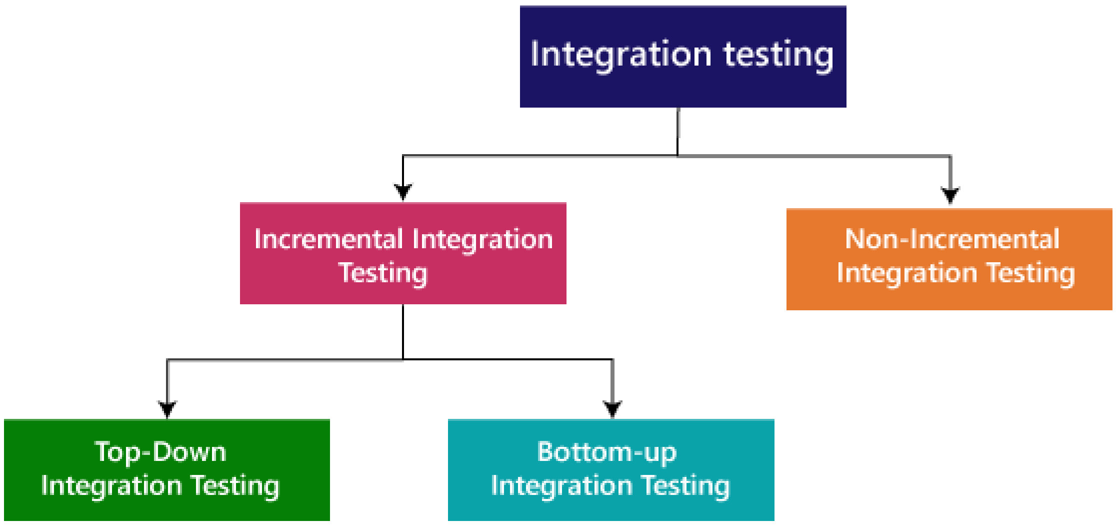
<u>Types of integration testing---</u>

Integration testing can be classified into two types—

- Incremental integration testing
- Non incremental integration testing

Again, incremental integration testing is classified into two types. These are

- Top-down incremental integration testing
- Bottom-up incremental integration testing


Integration testing
Incremental Integration Testing
Non-Incremental Integration Testing
Top-Down Integration Testing
Bottom-up Integration Testing

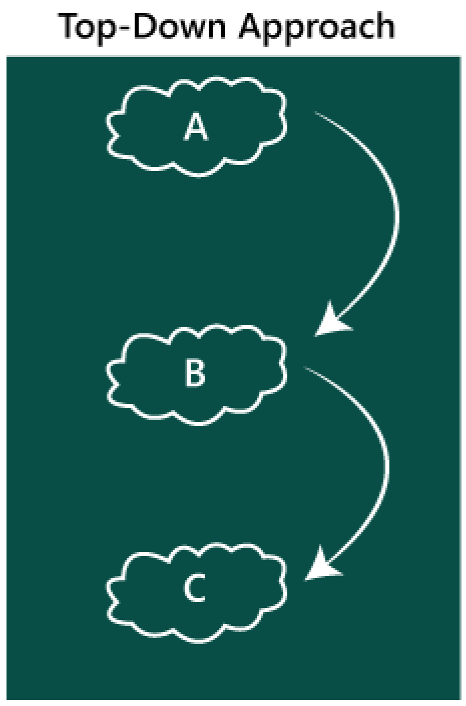
<u>Incremental integration testing---</u>

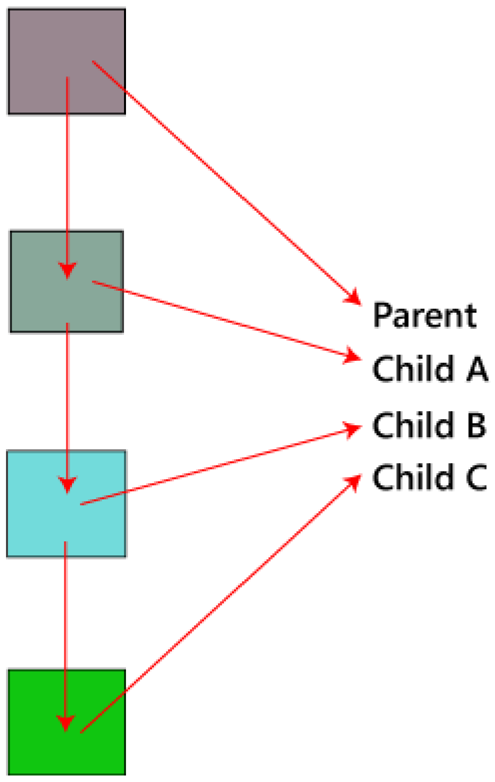
Incrementally adding the modules and testing the data flow between the modules is called as incremental integration testing.

In other words, when ever there is a clear relationship between modules, we go for incremental integration testing. Suppose we take two module and analysis the data flow between them if they are working fine or not. If these modules are working fine, then we can add one more module and test again. And we can continue with the same process to get better result.

Top-down incremental integration testing—

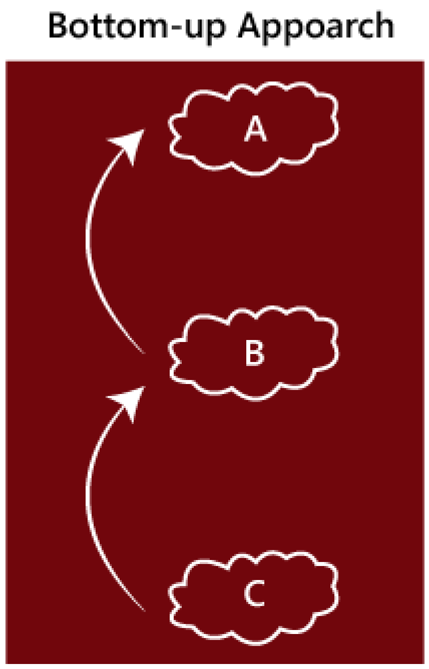
Incrementally adding the module and testing the data flow between the module and make sure that the module that you are adding should be the child of previous one.

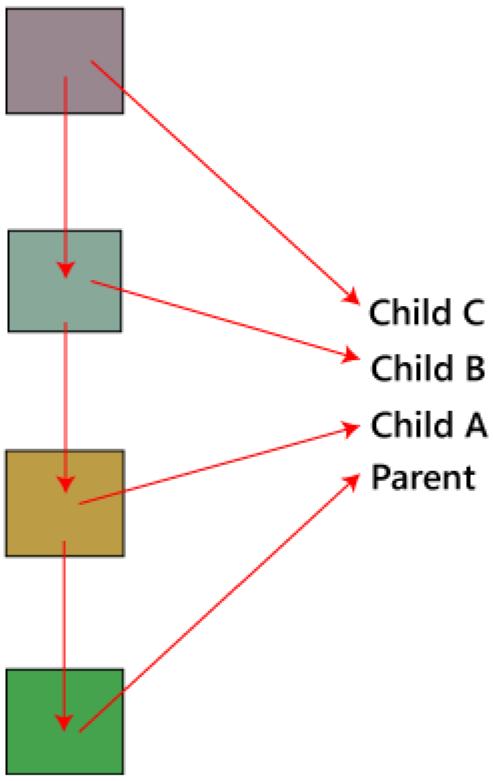




Bottom-up incremental integration testing—

Incrementally adding the modules and testing the data flow between the modules and make sure that the module That you are adding should be parent of previous one.

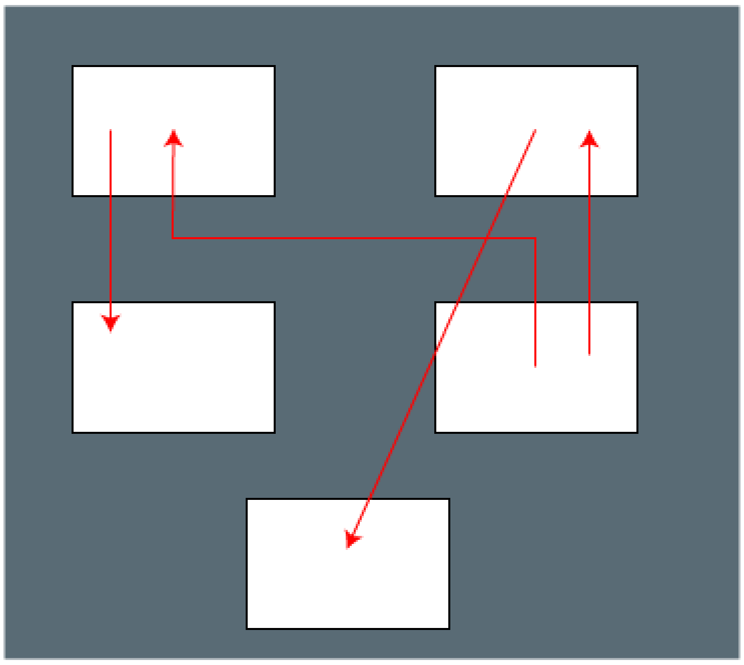


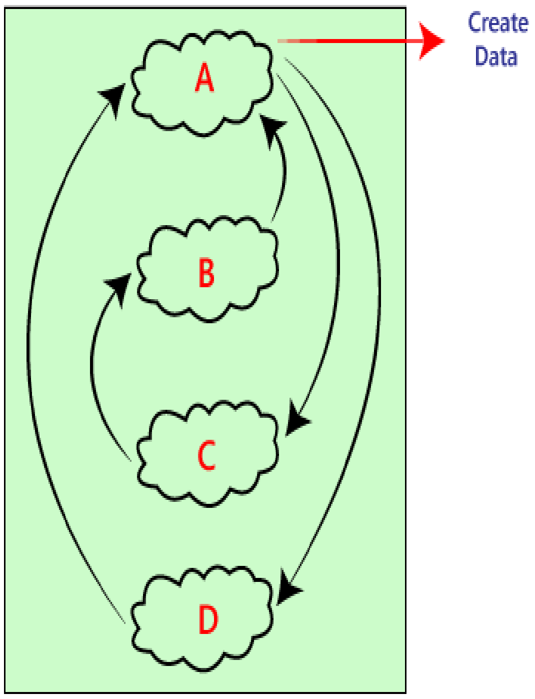


Note- in incremental integration testing data flow happens between one single module to another single module only. That is either from parent to child or child to parent.

Non incremental integration testing—

Whenever the data flow is complex and very difficult to classify a parent and child, we will go for the non-incremental integration approach. In this approach data flow happens from one single module to multiple other modules. The non-incremental integration method is also known as the big bang method.





Disadvantage of non-incremental integration testing—

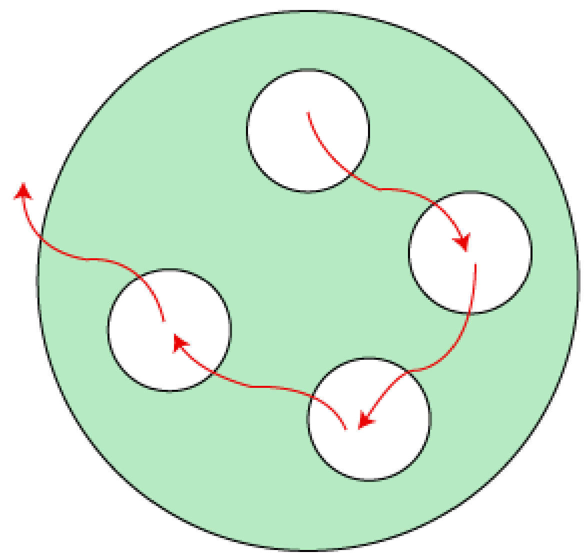
- Small modules missed easily.
- Root cause analysis is difficult that is identifying root cause of the defect is very difficult.

When we use non incremental integration testing---

Whenever there is a complexity in the data flow. When ever we are not sure about parent and child module.

iii. System testing----

It is an end-to-end testing wherein we navigate between multiple features of an application and we check whether the last feature is working fine or not.



Before we performing system testing, functionality and integration testing scenarios should be working fine.

In an application there will be multiple end to end scenarios and whenever we test those end-to-end scenarios, it becomes system testing or entire software testing.

Example---

Purchase a product from amazon.

$$\begin{tikzcd}
\text{Create an account} \arrow[d] \\
\text{Login to account} \arrow[d] \\
\text{Search for the product} \arrow[d] \\
\text{Add product to Wishlist} \arrow[d] \\
\text{Add product to cart} \arrow[d] \\
{}
\end{tikzcd}$$

Purchase the product

System testing is defined as process of end-to-end testing where in testing environment is similar to production environment.

Why testing environment is similar to production environment?

If both environments are not similar then chances are there software might get crash due to change in configuration.

Testing environment----

It is a setup used by testing team of the software organization in order to perform the testing on the product as soon as they get it from the development team.

Production environment---

it is a setup used by the customer in order to perform the testing on the product as soon as they get it from the software organization.

Test cycle: -

It is the time duration taken to start and complete the testing of one build, is called as one test cycle.

Respin: -

It is the process of getting multiple builds in a single test cycle. Respin occurs due to critical defects present in the application. If we get respin again and again in our testing project we should inform to the project manager.

Release: -

Time duration taken from requirement collection stage till the deployment stage, it is called as one release.

A software can have a multiple release. (Version of software)

Example—

1 release=45-50 test cycle

1 release=50 test cycle

Minimum no of builds---50no.

>50 builds----respin.

Patch: -

Patch is an addition of new codes into an existing software.

We can install this patch into the existing software.

A patch can contain security updates, addition of new features, bug fixes, modification etc.

iv. **Acceptance testing/user acceptance testing: ---**

It is an end-to-end testing done by the customer as soon as the customer gets the product from the software organization.

Acceptance testing is generally don in production environment to check whether the product can be acceptance or not.

<u>Who performs user acceptance testing/approach to do Acceptance testing: ----</u>

Approach 1: -

In this approach we will discuss how the acceptance testing is performed, and here the customer's test engineer will do the acceptance testing.

It is an end-to-end testing done by the testing team of the customer seating at customer's place where in they check whether all the critical end to end business scenario are working fine or not.

Example---

The blue dart provides the requirement, and tcs develops the application perform all the testing and handover to the blue dart company.

Now the question arises the blue dart will use the application as soon they get it from tcs? No.

The blue company has a group of test engineers, after they get the software and this team will start the testing the application. And this end-to-end testing is done at the customer environment, which is called the user acceptance testing.

Approach 2: -

In this case we will see how the employee become end users and performing acceptance testing.

It is an end-to-end testing done by the end users where in they use the bita version of the software and according to the uses, they can provide the necessary feedback.

Example—

The application is developed and tested at tcs environment and then sent to blue dart. In the blue dart they have fewer test engineers, so they can't do acceptance testing. So for this out of 300 employees of blue dart they will provide the application to the 30 employees and install the application to their systems and ask them to start using the application and find any defect or issue.

Now 30 employees will do the dummy implementation, which means they provide the data into the application and also written the data manually. Here the employees become the end user and also identify the bugs and issue while using the application.

Approach 3: -

When ever customer does not have testing team and no end user involvement.

It is and end to end testing done by the testing team of the software organization sitting at customer place where in they check whether all the critical end to end scenario are working fine or not.

Approach 4: -

It is an end-to-end testing done by the testing team of the software organization sitting at their own place where in they check whether all the critical end to end scenario are working fine or not.

Why customer does acceptance testing?

- To check whether the features requested by the customer are present or not.
- To check whether the features are working fine or not.
- Under business pressure software organization must push the software with lots of defect.

Why more rounds of acceptance testing will occur?

- If a customer identifies any critical defects in the application.
- Customer after getting the software gets new new idea and ask for lots of changes.

Stages of acceptance testing---

- Pre alpha stage--- in this stage only the front end of the application is developed.
- Alpha stage---backend logic of the application is written and testing team will start testing the software (black box testing)
- Beta stage—end users will use the beta version of the software and they will give the necessary feedback.
- Rc stage (release candidate) ---software organization will try to implement the feedback received from the end users.
- Release stage---actual software is delivered to the market.

Types of acceptance testing----

There are two types of acceptance testing. These are---

- **Alpha testing**
- **Beta testing**

Alpha testing	Beta testing
I. It is generally done by the testing team, who are internal employees of the organization.	I. It is generally done by the end user of the product.
II. Alpha testing required both white box testing and black box testing.	**II.** Beta testing use only black box testing.
III. To do alpha testing, a test environment is required.	**III.** No need to have testing environment.
IV. Alpha testing requires long execution cycle.	IV. Beta testing requires few weeks of execution cycle.
V. The developer can immediately fix the defects identified in alpha testing stage.	V. Developer can fix the defects of beta stage in the next release.

<u>Hot fix---</u>

It is the process of immediately fixing the critical defect identified by the customer or end users, while using the product in production environment. duration of the hot fix should be some hours.

<u>Root cause analysis---</u>

Root cause analysis is a process to find out the reason of the critical defect found in product and to perform root cause analysis, we use fish bone technique.

<u>Decision of root cause analysis---</u>

- What is the problem or defect.
- Sequence of events that led to the defect
- How long the defect existed in the software.
- What features were impacted due to the defect.
- What was the impact of the defect.

<u>Advantage of root cause analysis---</u>

- Prevents the defect from recurring (repeatedly)
- it reduces rework.
- Total investment will be less (save time and money)
- Promotes happy customers and end users.

Note—starting from hot fix till uploading the diagram in company website, entire time duration taken is called as incident management.

Steps to do root cause analysis—

I. Inform the root cause analysis team.
II. Define the problem.
III. Fix the problem by identifying the root cause.
IV. Implement root cause corrective action (Rcca)
V. Implement root cause preventive action. (Rcpa)

In order to perform root cause analysis, first we need to form a root cause analysis team which includes root cause analysis manager, development team, testing team, design team and maintenance team.

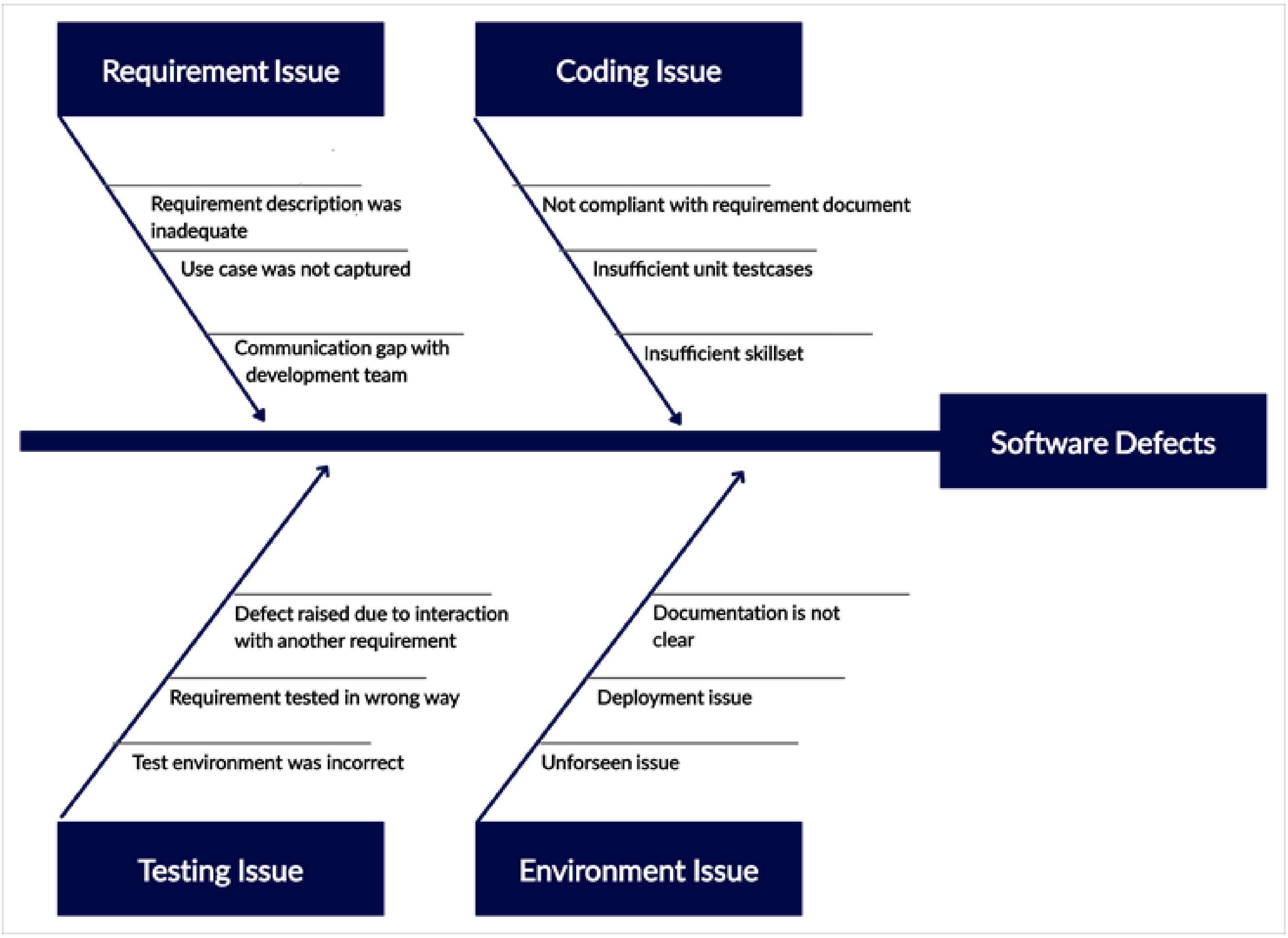
Once the root cause analysis team is formed, they will try to define the problem that is they will make certain discussion.

Once the discussion made, root cause analysis team can easily identify the root cause of the defect.

Once the root cause of the defect is identified, the root cause analysis team will implement root cause corrective action i.e., they will do hot fix.

Once hot fix is completed, the root cause analysis team will implement root cause preventive action where in they will come up with an approach so that the mistake do not appear again.

Steps to create fishbone diagram---


Requirement Issue
Requirement description was inadequate
Use case was not captured
Communication gap with development team
Coding Issue
Not compliant with requirement document
Insufficient unit testcases
Insufficient skillset
Software Defects
Defect raised due to interaction with another requirement
Requirement tested in wrong way
Test environment was incorrect
Testing Issue
Documentation is not clear
Deployment issue
Unforseen issue
Environment Issue

Fish bone diagram is a skeleton of a fish with problem or defect forming the head and causes of the defect forming the spine and bones.

Note—

Root cause analysis can be done using two techniques.

i. Fish bone technique
ii. 5 why technique

5 why technique—

- Why the defect was identified in the requirement stage?
- Why the defect was not identified in the design stage
- Why the defect was not identified in unit testing stage.
- Why the defect was not identified in testing stage?
- Why the defect was not identified in pre-production stage.

v. **Smoke testing/confidence testing/build verification testing/build acceptance testing-----**

Smoke testing is a testing technique in which we testing the basic or critical features of an application before doing proper or deep testing. If critical features are not working fine, we should stop or testing process. That is, we can not test the major and minor features.

In the smoke testing, we only focus on the positive flow of the application and enter only valid data, not the invalid data. In smoke testing, we verify every build is testable or not, hence it is also known as build verification testing.

Why we do smoke testing?

- We will do smoke testing to ensure that the product is further testable or not.
- While doing smoke testing, we are going to identify only critical defects, so we are helping

the development team to fix critical defect at an early stage.

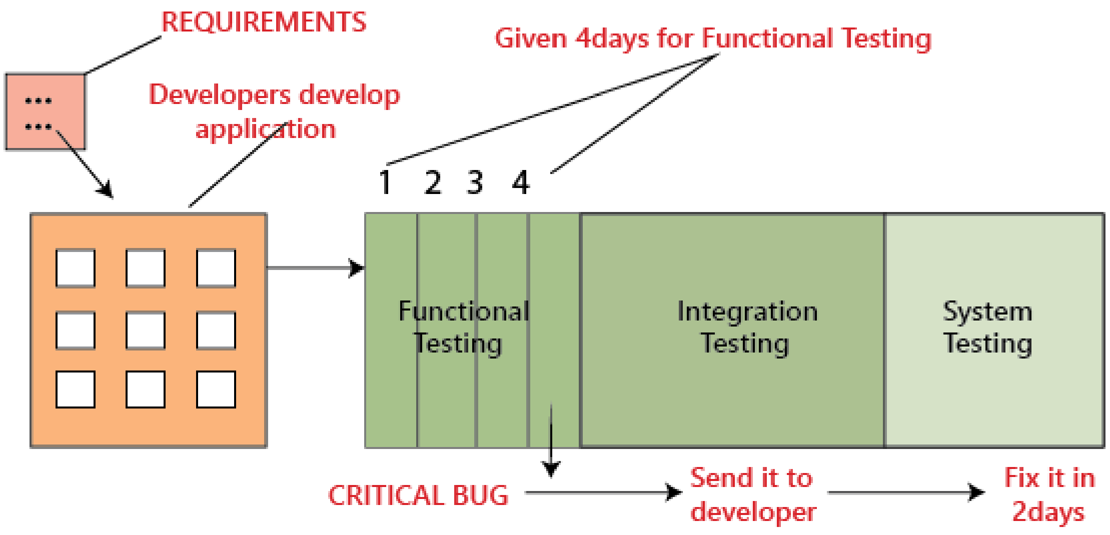
When we do smoke testing?

Generally smoke testing is done on initial build (new build) as soon as we get the build from the development team.

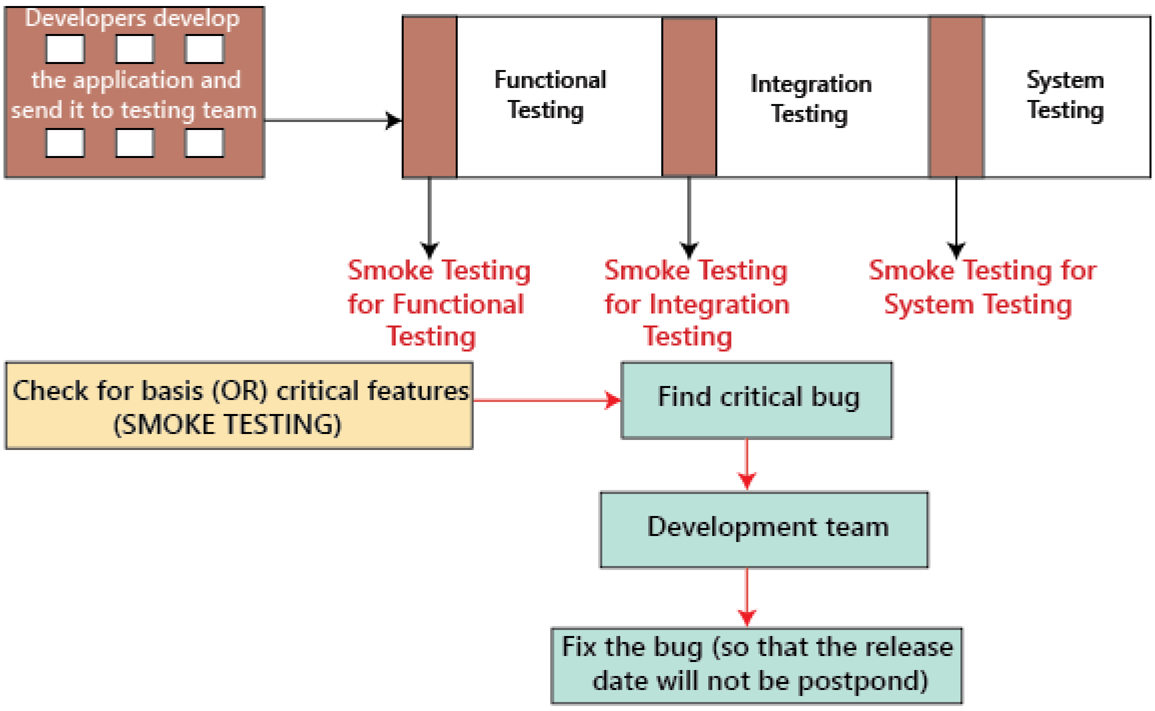
Example---

The developer develops the application and handed over to the testing team, and the testing team will start the functional testing

Suppose we assume that four days we are given to the **functional testing**. On the first day, we check one module, and on the second day, we will go for another module. And on the fourth day, we find a **critical bug** when it is given it to the developer; he/she says it will take another two days to fix it. Then we have to postpone the release date for these extra two days.



To overcome this problem, we perform **smoke testing**, let us see how it works, in the above situation, instead of the testing module by module thoroughly and come up with critical bug at the end, it is better to do **smoke testing** before we go for functional, integration and system testing that is, in each module we have to test for essential or critical features, and then proceed for further testing as we can see in the below images:


Developers develop the application and send it to testing team
Functional Testing
Integration Testing
System Testing
Smoke Testing for Functional Testing
Smoke Testing for Integration Testing
Smoke Testing for System Testing
Check for basis (OR) critical features (SMOKE TESTING)
Find critical bug
Development team
Fix the bug (so that the release date will not be postpond)

Can developer perform smoke testing?

Yes, developer while performing unit testing, they first check whether all the critical lines of code are working fine or not.

How to perform smoke testing

Smoke testing always start with giving valid data.

Smoke testing is a type of +ve testing. To do smoke testing we should follow the requirement document. In smoke testing we test only critical features, we don’t test major and minor features because time available is very less.

While doing smoke testing, we test the critical functionality, critical integration and critical system scenario.

Smoke testing is a kind of health checkup of a build. So that is why, it is called as build verification testing and build acceptance testing.

vi. **Sanity testing---**

Generally, sanity testing is done on stable builds and it is also known as a variant of regression testing. Here testing team will check, whether the new features added are working fine or not, modified features are working fine or not and bug fixes are done properly or not.

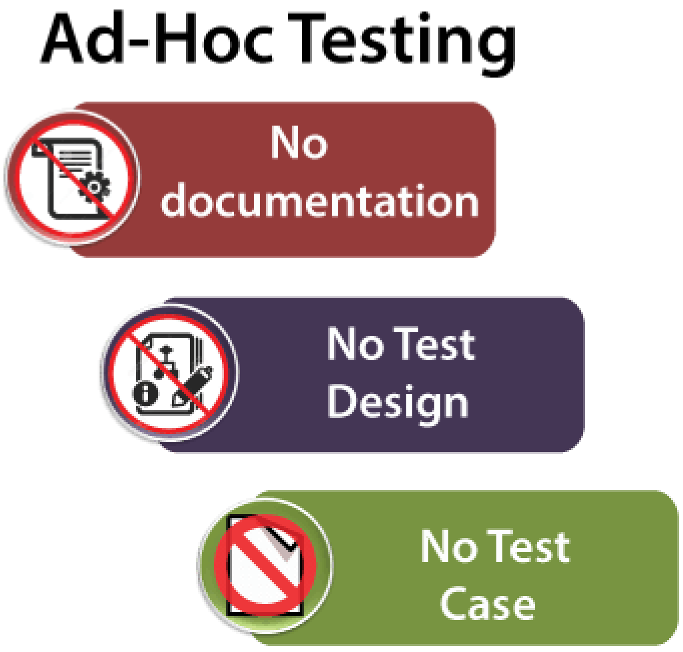
Different between smoke testing and sanity testing—

Smoke testing	**Sanity testing**
I. It is done on initial build.	I. It is done on stable build.
II. Test the basic or critical features of an application	II. We test whether the new features added are working fine or not, bug fix is done properly or not.
III. Smoke testing is done by both development team and testing team.	III. Sanity testing is only done by testing team.
IV. Smoke testing scenarios are documented.	IV. Sanity testing scenarios are not documented.

vii. Adhoc testing/random testing—

Adhoc testing is an informal or unstructured software testing.

Testing the application randomly without referring the srs document is called as adhoc testing.it is also called monkey testing and gorilla testing.
It is negative testing because we will test the application against the client's requirement.



Why we do adhoc testing?

Whenever end users get the software, they might use it randomly and find some issues and due to this a bad name might spread in the market due to identification of defect in live environment. This in turn will affect the business of the customer.

If we see the requirement document and test the product, number of defects we are going to catch will be less. So, in order to catch more and more number of defects, we should do adhoc testing.

When we do adhoc testing?

We go for adhoc testing when all type of testing are performed. Adhoc testing is generally done when we got free time.

In the early stage of product development number of defects we are going to catch will be more. So, at this time we should not do adhoc testing.

Advantage of adhoc testing---

- Adhoc testing cannot follow any process, that’s why it can be performed any time in the software development life cycle.
- The test engineer can test the application in their new ways that helps us to find out many numbers of bug as compared to the actual testing process.

- The developer can also execute the adhoc testing while developing the module that helps them to code in better way.
- When the in-depth testing is required in less time, adhoc testing can be performed and also deliver the quality product on time.
- adhoc testing does not require any documentation, that’s why tester can test the application with more concentration without worrying about the formal documentation.

Disadvantage of adhoc testing---

- adhoc testing is dependent on the test engineer’s product knowledge because he/she knows the flow of the application,so he/she knows where the application can collapse and a new test engineer may not that much familiar with the application.
- Sometime reproducing the bug is difficult because, in this testing, we did not follow any planning.

Different between smoke and adhoc testing----

Smoke testing

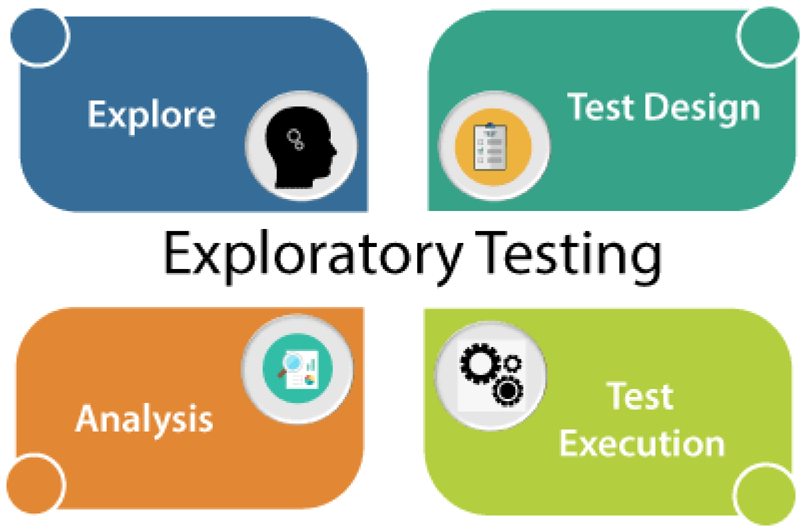
- Testing the basic or critical features of an application.
- We do smoke testing to check whether product is further testable or not.
- Smoke testing is done on initial build.
- Smoke testing is a positive testing
- To do smoke testing we should follow srs documents.

Adhoc testing

- Testing the application randomly.
- we do adhoc testing to find more defects.
- Adhoc testing is done whenever we have free time.
- Adhoc testing is a negative testing.
- We do not follow srs document.

viii. Exploratory testing--

In this testing we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application. This approach is known as exploratory testing.



Why and when we do exploratory testing---

- When ever we don’t have the requirement document.
- We have the requirement, but it is not understanding.
- When requirement document is there, but it is not understandable and not such time to go through the document.

Drawback of exploratory testing----

- Time consuming (it is a time taking process because we do not know the requirement, and which feature has to be tested first because we are just exploring the application.

- **The test engineer will misunderstand the feature as a bug.**
  For example, suppose we have one login page and requirement says we have to provide the necessary details like **username, password,** and **employee id** then click on the **login** button.
  But while performing exploratory testing, we only provide the details of **username, password,** and then click on the **login** button, but we have not entered the employee id. Since we don't have the requirement, and doing exploratory testing, that's why we feel that the employee id component is a bug, but it is a feature.

- **The test engineer will misunderstand the feature as a bug.**
  For example, suppose we have one login page and requirement says we have to provide the necessary details like **username, password,** and **employee id** then click on the **login** button.
  But while performing exploratory testing, we only provide the details of **username, password,** and

then click on the **login** button, but we have not entered the employee id. Since we don't have the requirement, and doing exploratory testing, that's why we feel that the employee id component is a bug, but it is a feature.

How to overcome the drawback of exploratory testing---

To overcome the drawback of exploratory testing we should frequently communicate with the following people-----

- Business analysist
- Development team
- Client.

ix. Regression testing----

Regression testing is nothing but testing the unchanged feature of an application to make sure that changes like adding a feature, deleting a feature or modification or fixing the defect is not impacting the unchanged feature of an application is called as regression testing.

Majority of time spent in testing is on regression testing.

When we do regression testing---

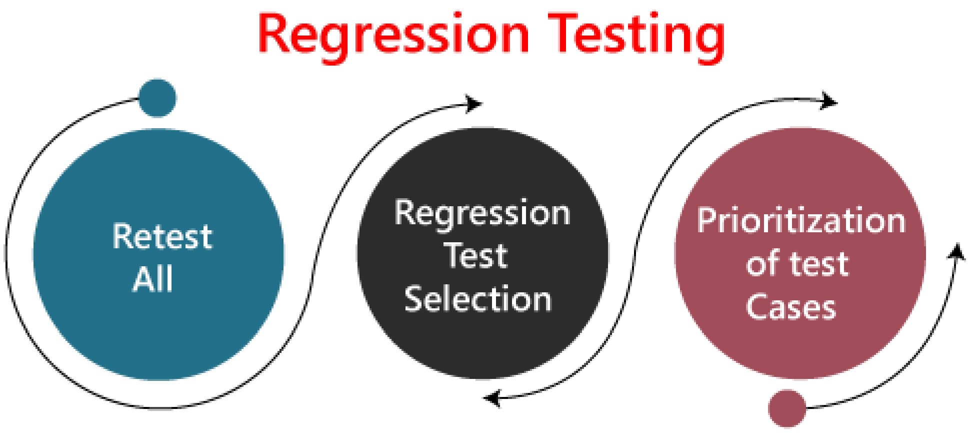
we can perform regression testing inn the following scenario, these are---

- Whenever new functionality added to the application.
- Whenever there is a defect/bug fix.
- Whenever there is a change in requirement.
- Whenever there is a platform/environment change.

How to perform regression testing------

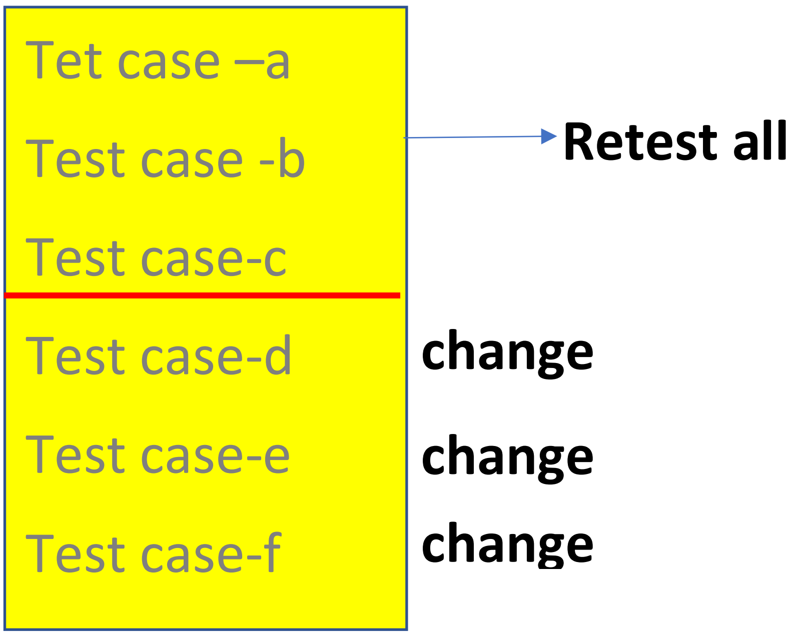
Regression testing can be performed using the following techniques.

I. Retest all
II. Regression test selection
III. Testcase prioritization



i. <u>Retest all----</u>

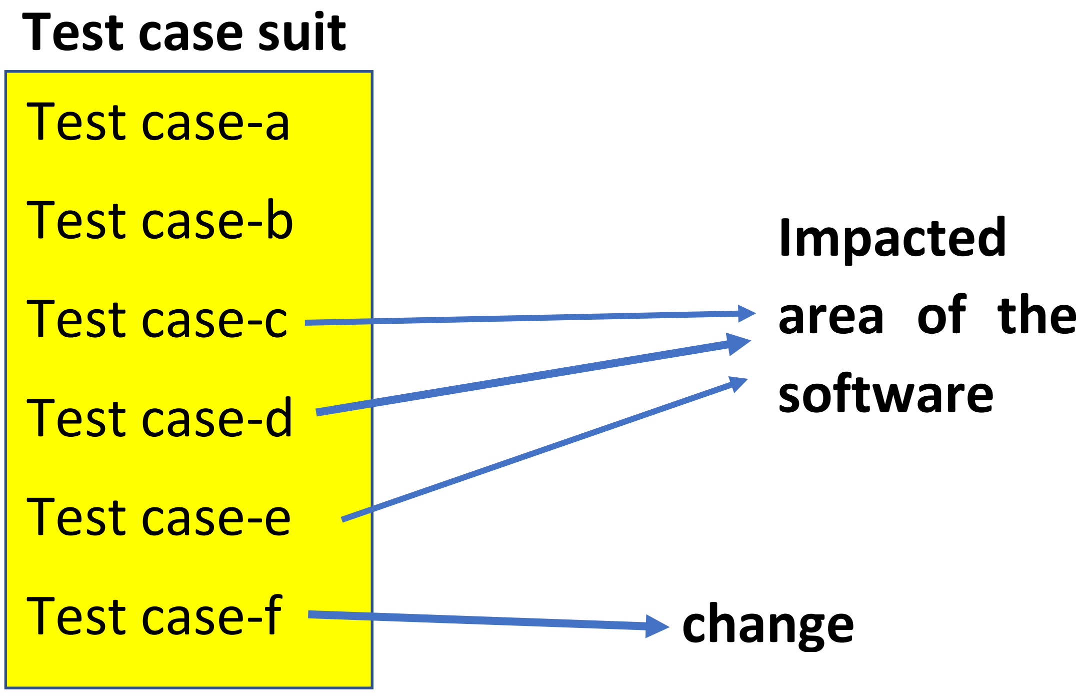
All the test cases in the existing test suit will be re-executed to ensure that there are no defect present.



This technique will be very expensive method as it needs more time as well as resources.

ii. <u>Regression test selection---</u>

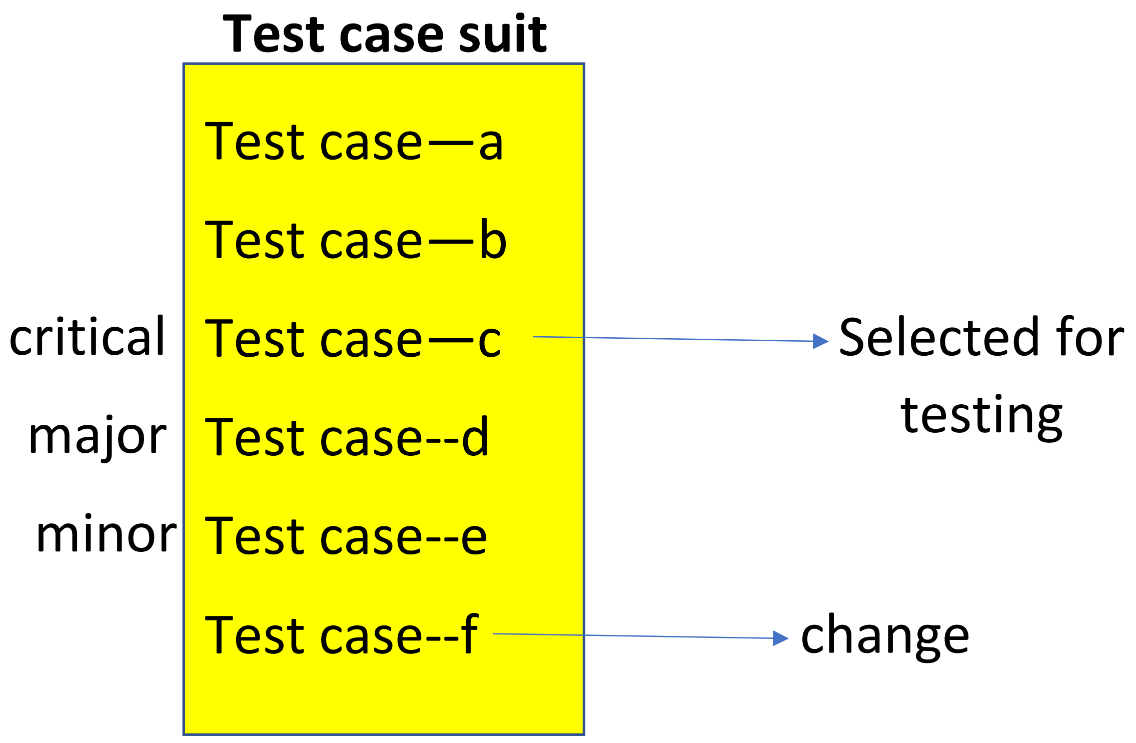
By using this technique, we select a part of test cases (only impacted test case) present in the test suit and we execute those test case instead of re executing all the test cases presented inside the test suit.



iii. Test case prioritization—

By using this technique, we select the test cases with high priority to get executed first instead of a medium and low priority.

Priority is decided on the basis of its impact on client's business.



Challenges in regression testing----

While performing the regression we may face the below challenges: ---

- Time consuming---
  Regression testing consumes a lot of time to complete. Regression testing involves existing tests again, so testers are not excited to re-run the test.
- Complex----
  Regression testing is complex as well. When there is a need to update any product, lists of the test are also increasing.
- Identify impact area.
- Test case increases release by release
- Less resources
- No accuracy
- Repetitive task

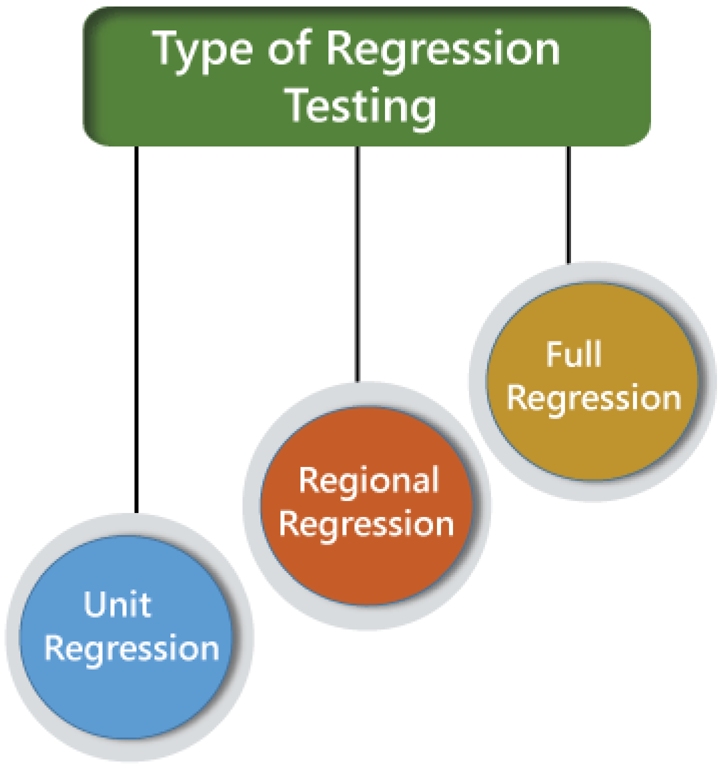
Best practice for regression testing---

- Select those test cases that causes frequent defects (prioritizing the text cases).
- Select the particular end to end scenarios (work flow) that has more visibility to the end users.
- Conduct regression test immediately after a smoke and sanity testing.
- Run the regression test case frequently to avoid release risk.
- Select the right automation tool to speed up the regression test processed.

<u>Types of regression testing----</u>

Based on changes, we should do different type of regression testing. These are---

I. Unit regression testing
II. Regional regression testing
III. Full or complete regression testing.



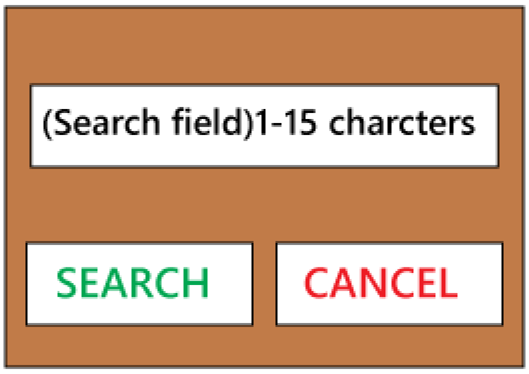
i. **<u>Unit regression testing</u>----**

In unit regression testing, we are going to test only the changed unit, not only the impact area.

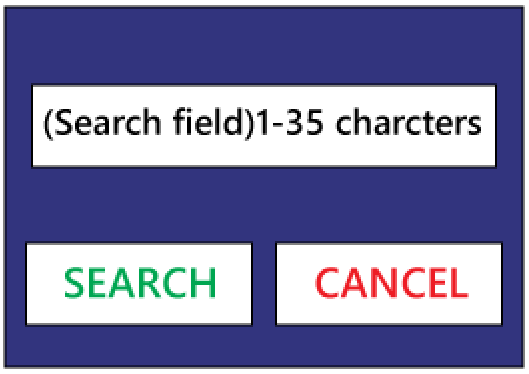
Example—

In the below application and in the first build, the developer develops the search button that accepts1-15 characters. Then the test engineer tests the search button with the help of the test case design technique.

Now the client does some modification in the requirement tool and also request that the search button can accept the 1-35 characters. The test engineer will test only the search button to verify that it takes 1-35 characters and does not check any further feature of the first build.



Build 1-B001



Build 1-B002

Therefor we can say that by testing only the changed features is called the unit regression testing.

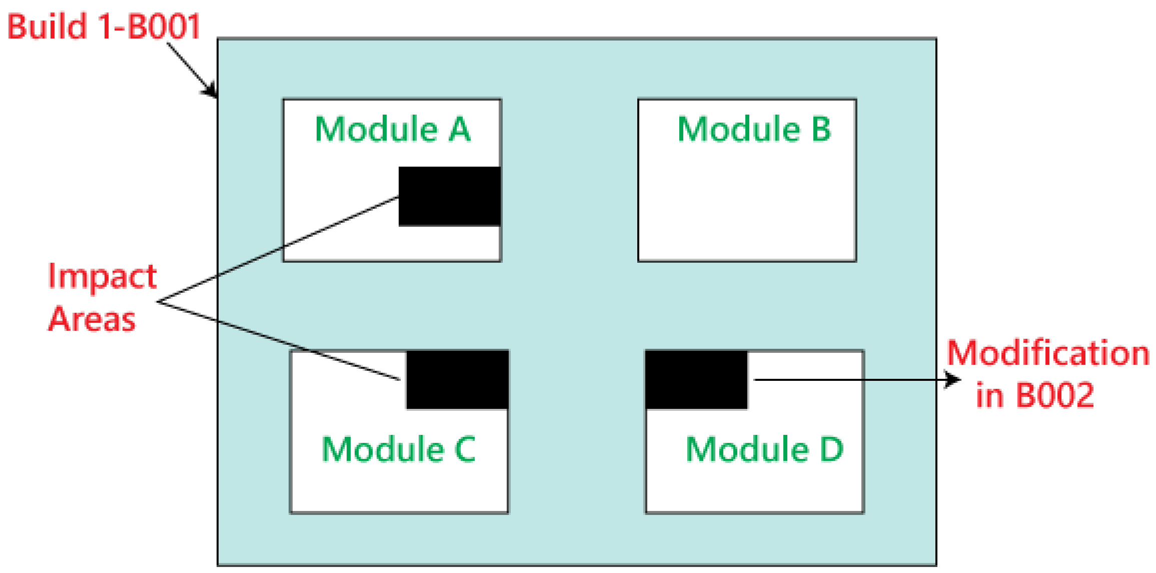
ii. **<u>Regional regression testing(rrt)—</u>**

In regional regression testing we are going to test the modification along with impact area or regions.

For example----

In the below image as we can see that we have four different modules, such as module A, module B, module C, and module D, which are provide by the developers for the testing during the first build.

Now the test engineer will identify the bugs in module D. the bug report is sent to the developers, and development team fixes those defects and sends the second build.



In the 2nd build, the previous defects are fixed. Now the test engineer understands that the bug fixing in module D has impacted same features in module A and module C. hence the test engineer first tests the module D where the bug has been fixed and then checks the impact areas in module A and module C. therefor, this testing is known as regional regression testing.

When we do regional regression testing, how we know that which are impact module. So before processing to the testing, there is a meeting which is called as impact analysis meeting. This meeting is conduct by test engineer or business analysist or product manager in which all developer, test engineer and product manager will sit together and they will decide which module are affect if developer do some changes in a module.

Disadvantaged of using unit and regional regression testing---

Following are some of the drawbacks of using unit and regional regression testing.

- We may miss some impact area.
- It is possible that we may identify the wrong impact area.

iii. **Full or complete regression testing(frt)—**

Testing the modified features and also the remaining part of the application is called the full regression testing.

When we perform full regression testing—

We will perform full regression testing when we have the following conditions.

- When the modification is happening in the source file of the product. For example—jvm is the root file of the java application, and if any change is going to happen in jvm, then the entire java program will be tested.
- When we have to perform n number of changes.

x. Compatibility testing---

Check the functionality/capability of an application on different software, hardware platforms, network and browsers is known as compatibility testing.

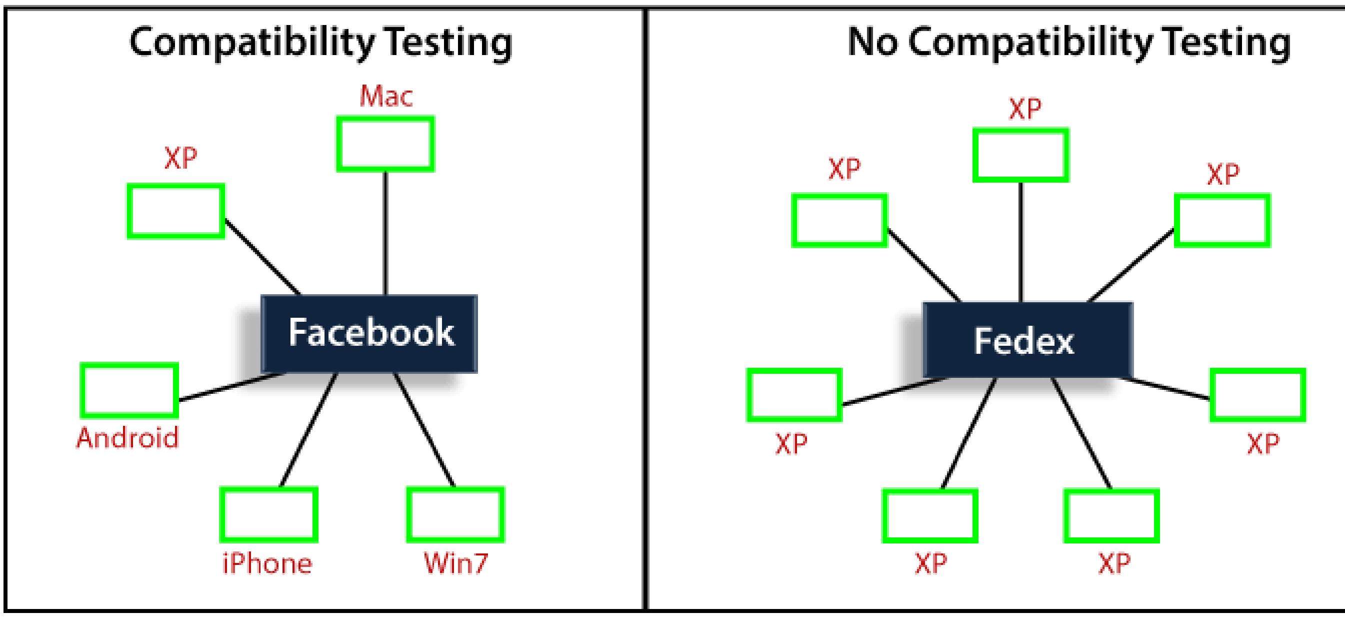
Compatibility testing is done only when the application becomes stable.

Why we do compatibility testing---

Whenever the product is delivered to the market, end user might use the product in different platform and they might get defect and due to this a bad name spreads in the market and number of end user will reduce which in turn will affect the business of the client. To avoid this issue, we do one round of compatibility testing.

When should we perform compatibility testing---

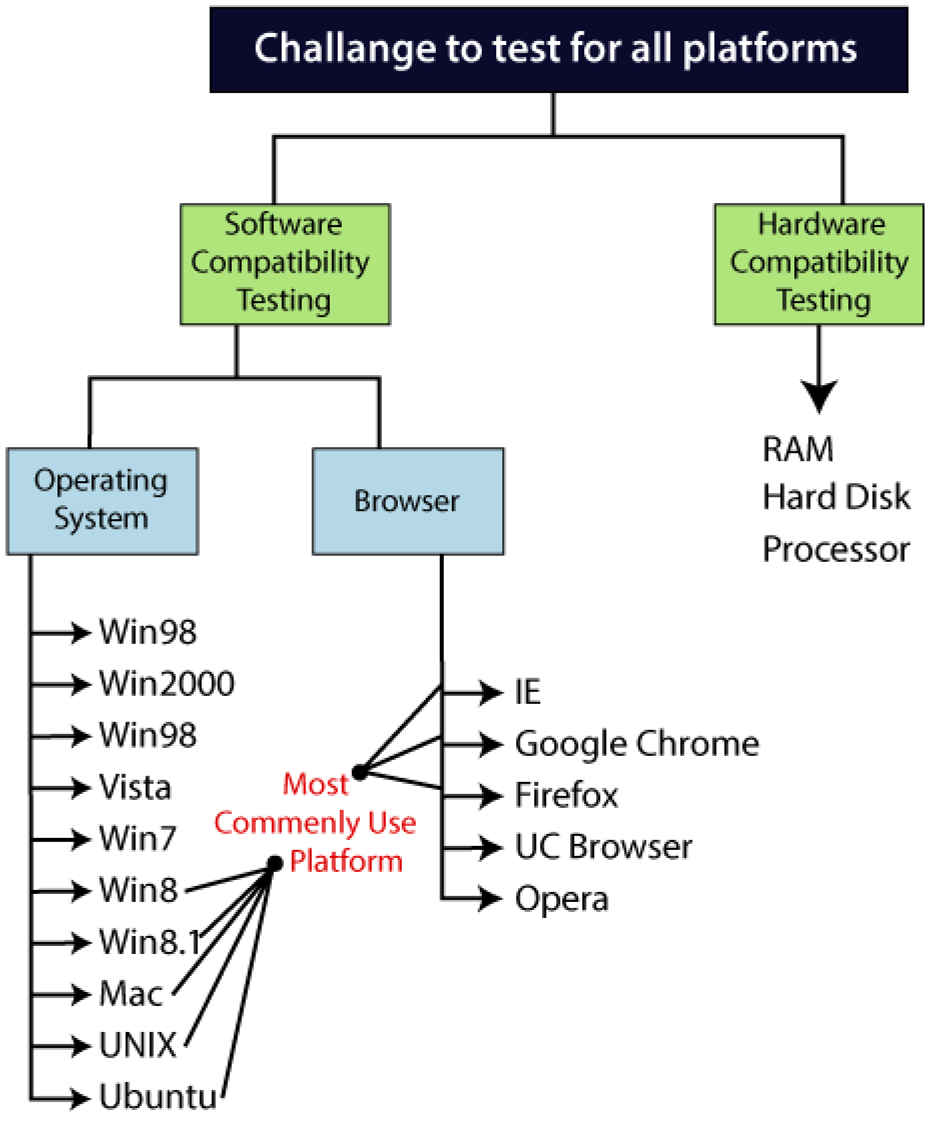
Generally, we go for compatibility testing only when the application or software is functionally stable.



How to decide platform as base platform---

we can’t test an application/software on all existing platforms since it will be a time-consuming process. Hence, we only select those platforms which are commonly used by the end users.

- Based on customer given platform.
- Based on maximum end users used platform.
- Development team, on which platform they developed the application, that platform we take as base platform.

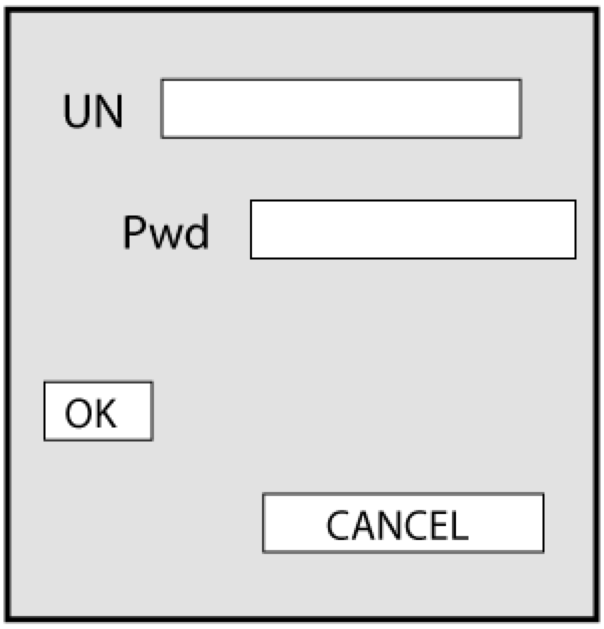


The various compatibility testing bug/issues are---

I. Alignment issue
II. Overlap issue
III. Scattered issue
IV. Look and feel issue
V. Broken frames issue
VI. Change in front size and color.

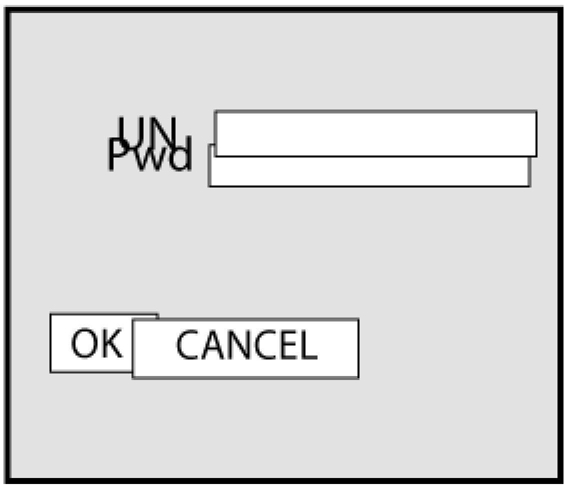
i. **Alignment issue---**

The alignment issue is that in which the element of the page is not aligned in a proper format as we can see in the below image.



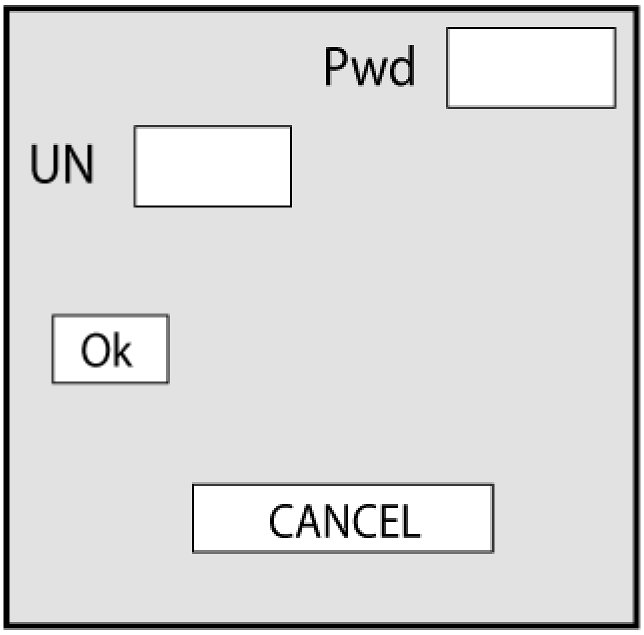
ii. **Overlap issue—**

When one attribute is overlapping to another attribute, it may happen when we are trying to open the application on the different platforms or browsers as we can see in the below image.



iii. Scattered issue----

When the test engineer performing compatibility testing on the application, and that application is not compatible with all browsers, and platform that's why the scattered issue may occur as we can see in the below image.



xi. Globalization testing---

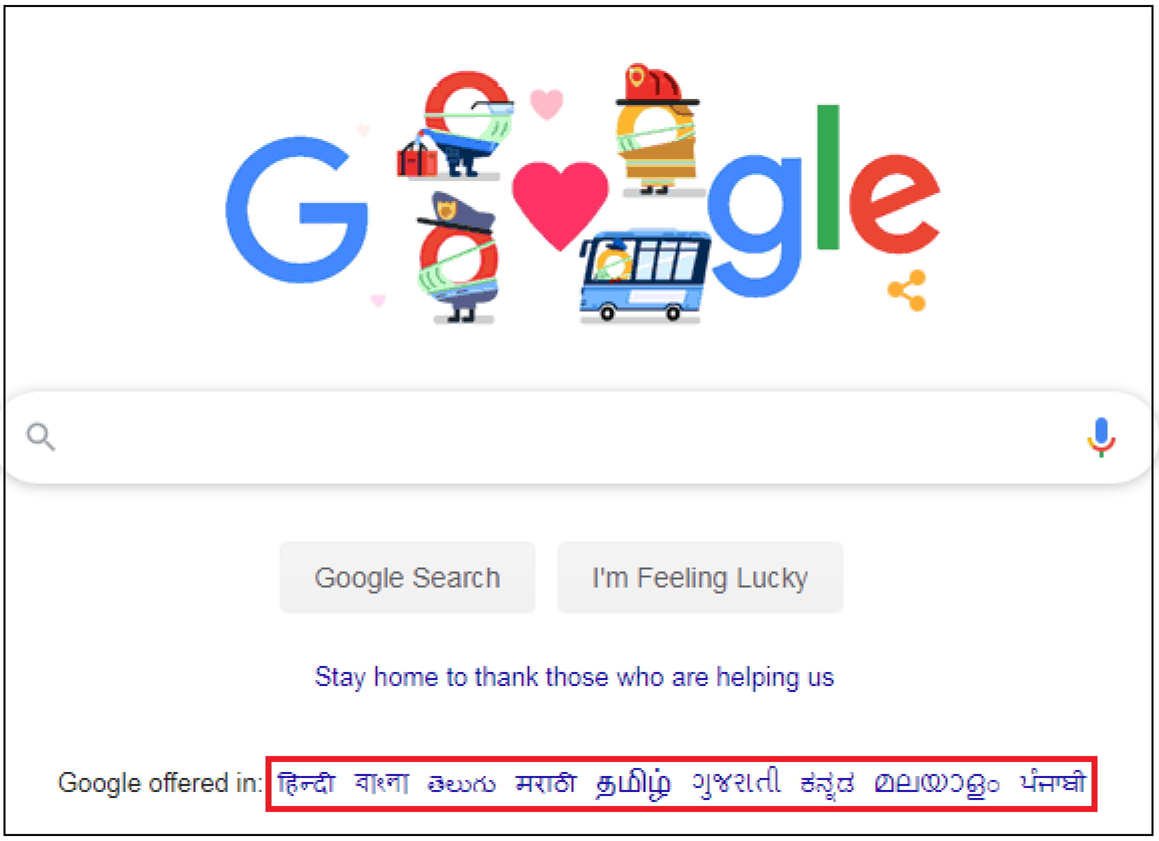
Whenever a software is developed in multiple language, it is called as a process of globalization.

Testing the software which is developed for multiple languages, it is called as globalization testing.

This testing ensure that the application will support multiple languages and multiple features.

For example---

In India, the google.com supports most of the languages, and it can also be retrieved by the large numbers of people of the various countries as it is a globalized application.

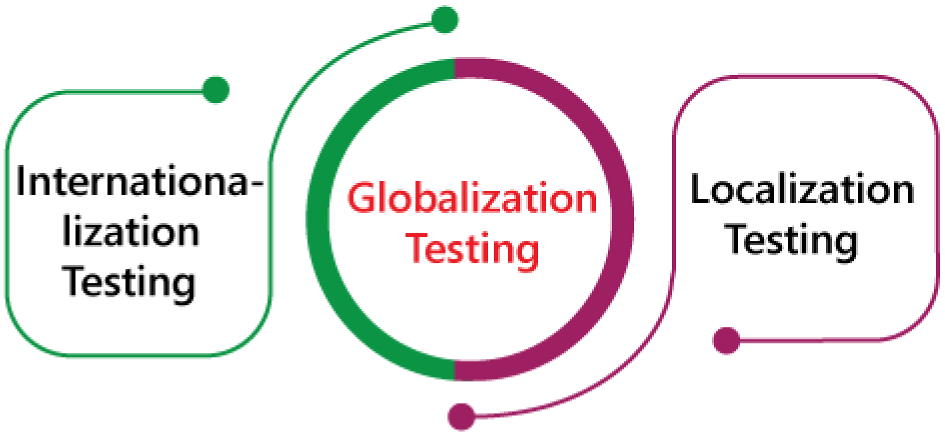

Google Search
I'm Feeling Lucky
Stay home to thank those who are helping us
Google offered in: हिन्दी বাংলা తెలుగు मराठी தமிழ் ગુજરાતી ಕನ್ನಡ മലയാളം ਪੰਜਾਬੀ

Purpose of globalization testing----

The primary purpose of globalization testing is as follows:

- It is used to make sure that the application is to support all the languages around the world.
- It is used to define the user interfaces of the software.
- This testing will focus on the world-wide experience of the application.

Types of globalization testing----



i. **Internationalization testing---**

The internationalization testing is the procedure of developing and planning the software or the application (product) or file content, which allows us to localize our application for any given language, culture, and region without demanding any changes in the source code. This testing is also known as I**18N testing,** and here **18** is the number between **I** and **N** in the **I**nternationalizatio**n** word.

I18N testing checklist (QA team)—

- Check whether product is working fine or not across different setting as per the country.
- Verify the installation processing using different setting.
- Check whether product is working fine or not across language setting and currency setting.

ii. **Localization testing---**

Here we check whether certain features are localized under country standard or not.

Example—

Facebook ----local languages (Marathi, Bengali, kanada, Telegu etc….)

It increases their market value/business (attract mor end user).

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

In L10N testing, we need to check whether all the functionalities are working as per the local settings like pin code, Bank IFSC code, vehicle registration number, regional language etc.

Nonfunctional testing

Nonfunctional testing is done to verify the nonfunctional requirement of the application like performance, usability, accessibility etc.

The best example of nonfunctional test would be to check how many people can simultaneously login into a software.

Functional and nonfunctional testing both are mandatory for newly developed software. Functional testing checks the correctness of internal functions while non-functional testing checks the ability to work in an external environment.

Parameters to be tested under nonfunctional testing-

1. Performance testing
2. Usability testing
3. Reliability testing
4. Recovery testing

5. Accessibility testing

1. Performance testing—

Testing the stability and response time of an application by applying load is called performance testing.

Stability means ability to withstand when n number of users using the application simultaneously for a particular time.

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

Load means that when n number of users using the application simultaneously or sending the request to the server at a time.

Types of performance testing---

- Load testing
- Stress testing
- Volume testing
- Soak testing

Load testing---

The load testing is used to check the performance of an application by applying some load which is either less then or equal to the desired no of users.

For example: -

The below image, 100users 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.

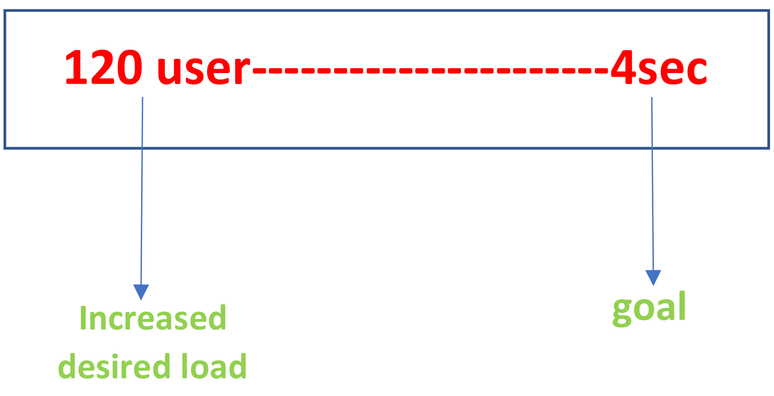
100 users—3 secs

Stress testing: -

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

Example-

If we took the above example and increased the desired load 100 to 120 users and the goal is 4sec.while performing the stress testing in this scenario, it will pass because the load is greater(100up) then the actual desired load.



Volume testing-

In volume testing we check the behavior of an application by inserting large volume of data to it.

Volume is a capacity while load is quantity, that is load testing means number of users, and volume testing means amount of data.

Soak testing-

In soak testing we will check the behavior of an application by applying load continuously for a particular period of time.

2. Usability testing—

Testing the user friendliness, efficiency and accuracy of an application is called as usability testing.

Why we do usability testing---

- ✓ To check whether frequently used features are easily accessible or not.
- ✓ To check whether frequently used features are displayed either in left or top navigation bar.
- ✓ To check whether the application has been developed as per the characteristics of a user-friendly software or not.

When we do usability testing---

Usability testing is done in the prototype stage only.

<u>Characteristics of user friendly software------</u>

- ✓ <u>Easy to understand</u>

  All the feature of software or application must be visible to the end users.
- ✓ <u>Easy to access</u>

  A user-friendly application should be accessible by everyone.
- ✓ <u>Look and feel</u>
- ✓ <u>Faster to access</u>

  The software should be faster while accessing which means that the application's response time is quick. If the response time is slow, it might happen that the user got irritated. We have to ensure that our application will be loaded within 3 to 6 seconds of the response time.

✓ Effective navigation

Good internal linking.
Informative header and footer.
Good search feature.

✓ Good error handling

Handling error at a coding level makes sure that the software or application is bug free and robust.

By showing the correct error message will help to enhance the user experience and usability of the application.

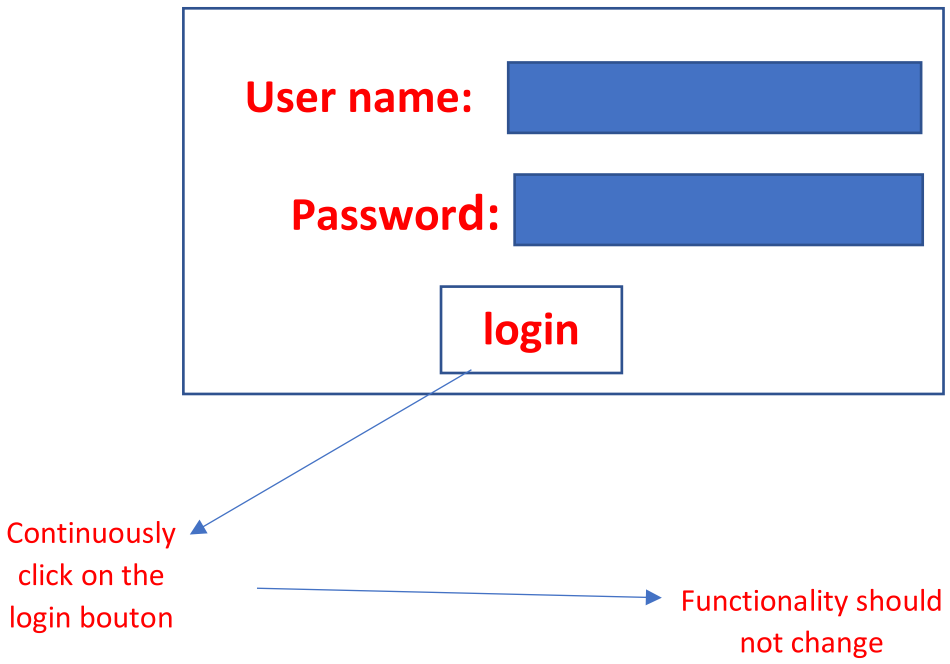
3. **Reliability testing—**

In reliability testing, testing the functionality of an application continuously for a particular period of time.

For example—

Let us consider in our mobile phone the software may not work continuously. In mobile, after a week or ten days, the phone may hang up because whatever feature is present in the phone, whenever it is used it continuously creates an object in the RAM of the phone. Once the RAM is completely filled with objects, if we get nay call or message, we will be unable to press the call button and the phone hangs up. Thus, we make use of a cleanup software. When we switch off the phone and switch it on again, all the objects in the RAM get deleted.

Example—


User name:
Password:
login
Continuously click on the login bouton
Functionality should not change

4. Recovery testing—

Recovery testing is a testing where the test engineer will test the application to check how well the application/software recovers from disasters or crash.

Example—

Suppose we are using the browser, let say google chrome and the power goes off. when we switch on the system again and 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.

Some of the most common failures which need to test for recovery: -

- External device not responding
- Network issue
- Wireless network signal loss
- Power failure
- Server not responding
- Database overload

- External server not reachable.
- Physical condition
- Stopped services.

Steps to perform recovery testing—

I. Introduce defect and crash the application (by applying load, by introducing lots of critical defects)
II. Open the task manager and kill the task/end the process (task gets removed from the task bar)
III. Uninstall the application and reinstall it (application should open with default setting)

5. Accessibility testing/ADA testing (American disability act)/508 compliance testing-----

Accessibility testing is used to test the application whether it can be easily accessed or can be easily used or not by physically challenged person. Here the physical disability could be old age, hearing, color blindness, eye sight problem, literacy disability (reading problem) etc.

It is also known as 508 compliance testing. In this, we will test a web application to ensure that every user can access the website.

Rules: --

- We should not use red and green color.
- Speech recognition software should be installed (convert the speech into text)
- Screen magnification software should be installed (used to enlarge the screen)
- Special keyboard should be installed (for easy typing)

TEST CASE

Test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirement or not.

Test case is a set of step-by-step instructions to verify something behaves as it is expected.

Why we write test cases?

- Test case gives detailed information about testing strategy, testing process, precondition and expected output.
- To have consistency in test case execution--------seeing the test case and testing the product.
- To avoid training every new engineer on the product ----

  When a test engineer leaves, he leaves with lot of knowledge and scenarios. Those scenarios should be documented, so that new engineer can test with the given scenarios and also write new scenarios.
- To depend on a process rather then a person.

When do we write test case?

Generally, we will write the test case whenever the developer is busy in writing the code.

We will write the test case when we get the following---

- When the customer gives the business needs then(srs documents), the developer starts developing and says that they need 3.5months to build this product.

- And in the meantime, the testing team will start writing the test cases.
- Once it is done it will send it to the test lead for the review process.
- And when the developers finish developing the product, it is handover to the testing team.

Different between test scenario and test case—

TEST SCENARIO	TEST CASE
I. It is a high level one line information about what to test in the application.	I. It is a detail explanation of how to test the application. It provides information about the testing strategy, testing process, pre conditions, and expected output.
II. These are high level action.	II. These are low level action.
III. Test scenario are derived from srs document.	III. Test cases are derived from the test scenario.
IV. Test scenario generally take less time to write.	IV. Test cases generally take more time to write.
V. Test scenario are easy to maintain due to their high-level design.	V. Test cases are hard to maintain.

vi. the test scenarios will help us in an agile way of testing throughout the functionality.	vi. the test case will help us in depth testing of the application.

Test case template: ----

test case name: ---

test case type: ---

module name: --

severity (critical/major/minor: ---

pre-condition: ---

Brief description: ---

HEADER

Serial number	Action (description)	input/ test data	Expected result	Actual result	status	comments

BODY

Author's name---

Reviewer's name: ----

Approved by: ---

Approval date: ----

FOOTER

Who write the test cases: ----

Test cases are generally written by the test engineers by referring the test scenarios that has been derived from the requirement document.

Rules and lessons to follow while writing test cases:

I. While writing the test cases, we should start from navigational steps.
    - Open the browser and enter the URL of the application.
    - Enter valid user name and password and click on login button.

II. While writing the test cases, we should use words like should be or must be.

III. While writing the test cases, we should imagine the application.

IV. If we organize our steps properly, number of steps will reduce.

Example: -----

Huge steps

a. Open the browser
b. Enter the url of the application
c. Enter valid user name.
d. Enter valid password.
e. Click on login button.

Less steps

a. Open the browser and enter the url of the application.
b. Enter valid user name and password and click on login button.

Types of test cases: -----

I. Functionality test case
II. Integration test case.
III. System test case
IV. Acceptance test case
V. Smoke test case (critical functionality/critical integration/critical system test cases.
VI. Regression test case (all the test cases that are executed again and again.

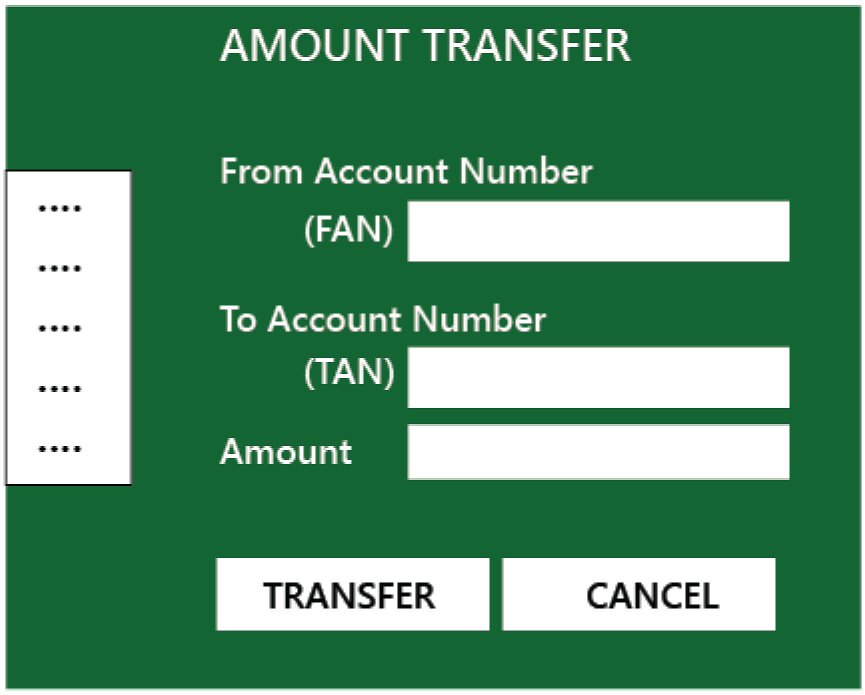
1. Functionality test cases: ---

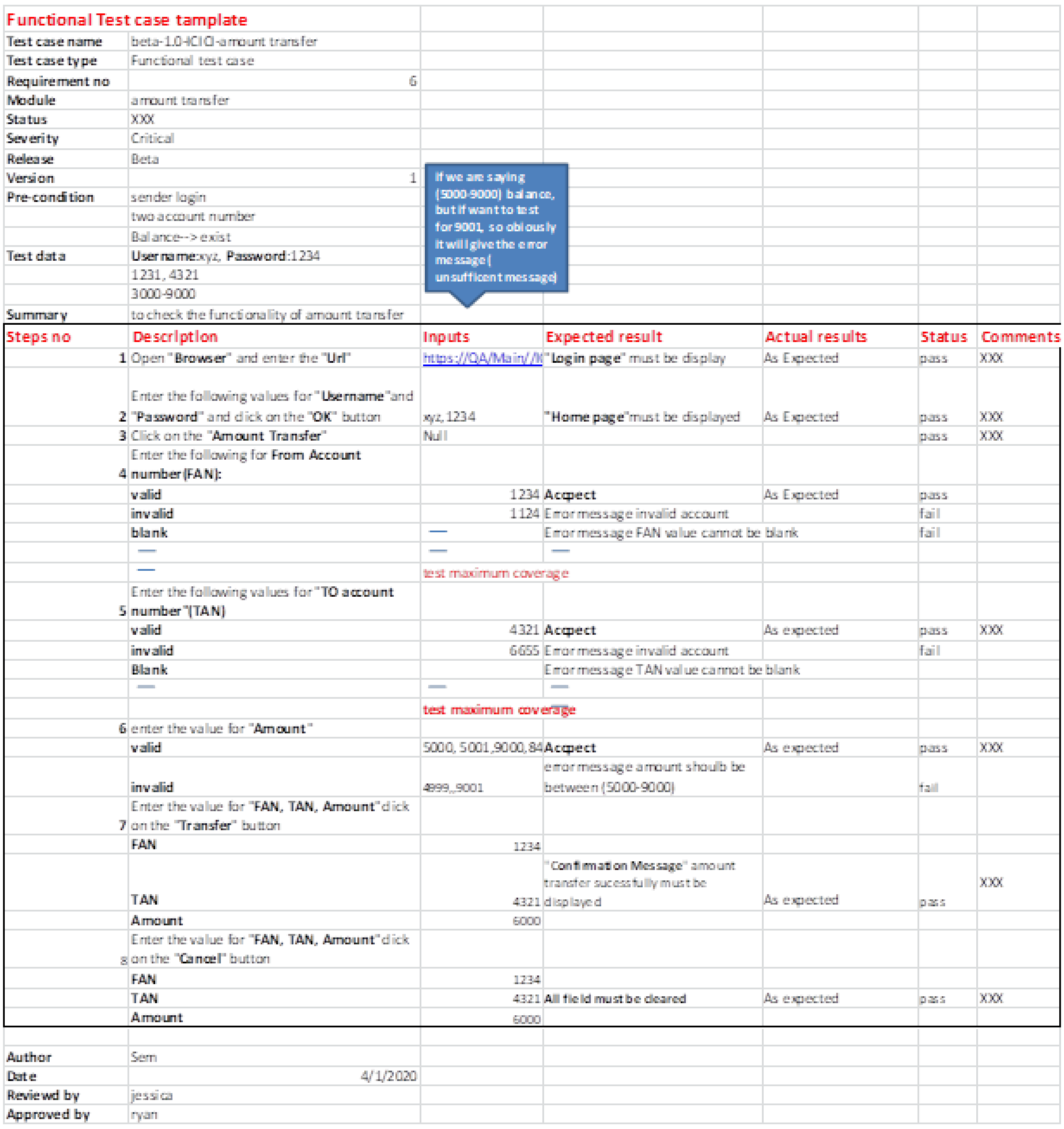
First, we check for which field we will write test cases and then describe accordingly.

Rules to write functional test cases: --

- In the expected results column, try to use should be or must be.
- Highlight the object names.
- We have to describe only those steps which we required the most, otherwise we do not need to define all the steps.
- To reduce the excess execution time, we will write steps correctly.
- Write a generic test case, do not try to hard code it.

Let say it is the amount transfer module, so we are writing the functional test cases for it and then also specifies that it is not a login feature.


AMOUNT TRANSFER
....
....
....
....
....
From Account Number
(FAN)
To Account Number
(TAN)
Amount
TRANSFER
CANCEL



Functional Test case tamplate						
Test case name	beta-1.0-ICICI-amount transfer					
Test case type	Functional test case					
Requirement no	6					
Module	amount transfer					
Status	XXX					
Severity	Critical					
Release	Beta					
Version	1					
Pre-condition	sender login					
	two account number					
	Balance--> exist					
Test data	**Username**:xyz, **Password**:1234					
	1231, 4321					
	3000-9000					
Summary	to check the functionality of amount transfer					
Steps no	**Description**	**Inputs**	**Expected result**	**Actual results**	**Status**	**Comments**
1	Open "**Browser**" and enter the "**Url**"	https://QA/Main/ /I(	"**Login page**" must be display	As Expected	pass	XXX
2	Enter the following values for "**Username**" and "**Password**" and click on the "**OK**" button	xyz, 1234	"**Home page**" must be displayed	As Expected	pass	XXX
3	Click on the "**Amount Transfer**"	Null			pass	XXX
4	Enter the following for **From Account number(FAN):**					
	valid	1234	**Accpect**	As Expected	pass	
	invalid	1124	Error message invalid account		fail	
	blank	—	Error message FAN value cannot be blank		fail	
	—	—	—			
	—	test maximum coverage				
5	Enter the following values for "**TO account number**"(**TAN**)					
	valid	4321	**Accpect**	As expected	pass	XXX
	invalid	6655	Error message invalid account		fail	
	Blank		Error message TAN value cannot be blank			
	—	—	—			
		test maximum coverage				
6	enter the value for "**Amount**"					
	valid	5000, 5001,9000, 84	**Accpect**	As expected	pass	XXX
	invalid	4999,,9001	error message amount shoulb be between (5000-9000)		fail	
7	Enter the value for "**FAN, TAN, Amount**" click on the "**Transfer**" button					
	FAN	1234				
	TAN	4321	"**Confirmation Message**" amount transfer sucessfully must be displayed	As expected	pass	XXX
	Amount	6000				
8	Enter the value for "**FAN, TAN, Amount**" click on the "**Cancel**" button					
	FAN	1234				
	TAN	4321	**All field must be cleared**	As expected	pass	XXX
	Amount	6000				
Author	Sem					
Date	4/1/2020					
Reviewd by	jessica					
Approved by	ryan					

If we are saying (5000-9000) balance, but if want to test for 9001, so obiously it will give the error message(unsufficent message)

2. Integration test case----

In this, we should not write something which we already covered in the functional test cases, and something we have written in the integration test case should not be written in the system test case again.

Rules to write integration test cases

- Firstly, understand the product
- Identify the possible scenarios
- Write the test case based on the priority

When the test engineer writing the test cases, they may need to consider the following aspects:

If the test cases are in details:

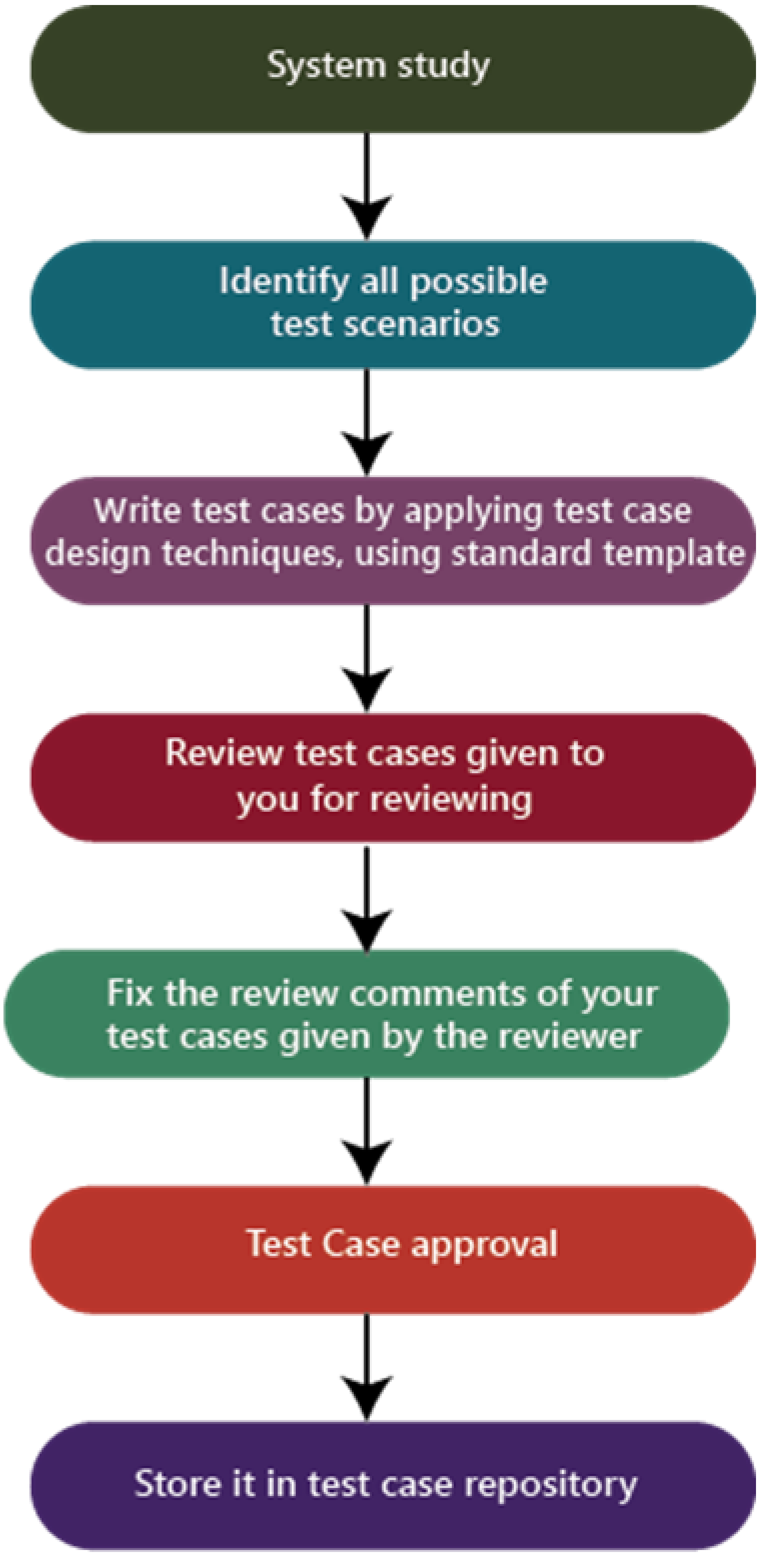
- They will try to achieve maximum test coverage.
- All test case values or scenarios are correctly described.
- They will try to think about the execution point of view.
- The template which is used to write the test case must be unique.

3. System test cases—

We will write the system test cases for the end-to-end business flows. And we have the entire modules ready to write the system test cases.

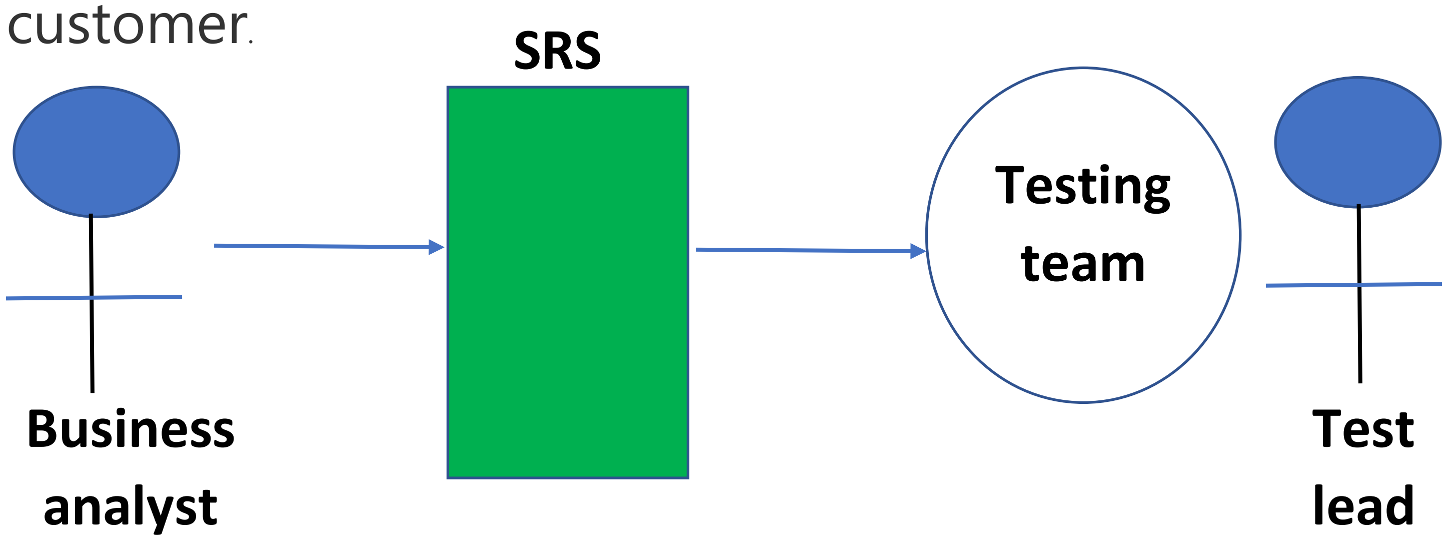
The process to write test cases--

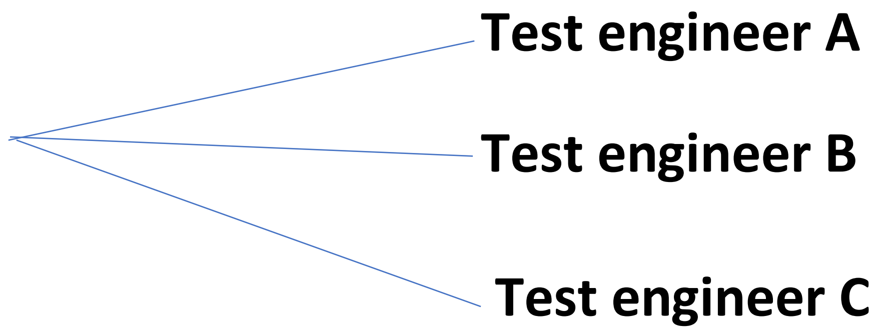
The method of writing a test case can be completed into the following steps, which are as below-


System study
Identify all possible test scenarios
Write test cases by applying test case design techniques, using standard template
Review test cases given to you for reviewing
Fix the review comments of your test cases given by the reviewer
Test Case approval
Store it in test case repository

System study--

in this, we will understand the application by looking at the requirements or the SRS, which is given by the customer.





Test engineer as soon as they get the copy of the srs document, they will go through the requirement document and they will try to understand the requirement of the customer.

While going through the requirement document chances are there, they might have some doubts/queries in understanding the requirement of the customer.

If in case they have any doubt, they should communicate with the following people---

- Business analyst
- Development team
- Test lead---arrange a conference call with the client/customer.

Identify all possible scenarios----

Once we have understood the requirement of the customer, test engineers will start identifying all the possible scenarios by referring the srs document. They are going to identify—

- Functionality scenario (valid and invalid)
- Integration scenario (data flow)
- System scenario (end to end)

Each and every test engineers will give a presentation of their derived scenarios.

Advantage---

Easily identify all the wrong, missing and repeated scenarios.

Prioritized identified scenarios---

Once the test engineers have identified all the possible scenarios, they need to prioritize all the identified scenarios on the basis of its impact on the business of the customer.

All the test scenarios are prioritized as---

- Critical scenarios (selected for smoke testing)
- Major scenarios
- Minor scenarios

Write test cases---

Author will start writing the test case based on the modules allocated to him/her.

- Critical test scenarios----critical test cases
- Major test scenarios---major test cases
- Minor test scenarios-----minor test cases

Generally test engineers will write the following test cases:-

- Functionality test case
- Integration test case
- System test case

Review test cases---

Review is generally done by another test engineer and we call him/her as the reviewer. Reviewer will review the test case of author to find out the mistakes in author’s test case.

Once reviewer identifies any mistakes in author’s test case, he will prepare a report which contains a list of all the mistakes done inside the test cases.

Why we review test cases---

- To find missing scenarios, wrong scenarios and repeated scenarios,
- To check whether the test case is easily understandable or not, so that any new engineer comes he can able to execute those test cases without asking any questions.
- To check whether all the attributes are covered or not.
- To check whether all the attributes contain relevant data or not.
- To check whether standard test case template is followed or not.

Which test case is called as a very good test case----

- It should be cover all the possible scenarios.
- It should be easily understandable.

On what basis test lead assigns review job to another test engineer?

- One who is very good in domain.
- One who known the product very well.
- One who is responsible, even though he is new to the feature, he can understand the feature and test it.

What are review ethics? (Ethics means norms and standards) ---

- Always review he content and not the author.
- While reviewing, spend time in identifying the mistake and not the solutions for it.
- Even after review if there are mistakes, then both author and reviewer are responsible.

Review comments report---

This report is generally prepared by the reviewer and the report contains list of mistakes done by the author in the test case. And the report is sent back to the author so that author can easily come to know where exactly the mistakes are present inside the test cases and rectify all the mistakes.

Serial no.	Test case name	comments	severity	author
1	SBI amount transfer (header)	Test case type attribute is missing	Major	fixed
2	SBI amount transfer (body—step no 22)	The scenario has been repeated	Critical	Fixed

3	SBI amount Transfer(footer)	Author’s name is missing	major	fixed

Fix review comments----

It is generally done by author where in he/she will fix all the mistakes done in the test case by referring the review comments report.

Once all the mistakes are fixed, author will generate a new/modified copy of the test case and send it back to the reviewer for rechecking.

Verify the fix---

it is generally done by the reviewer where in he/she checks whether the previous mistakes present inside the test cases has bee rectified or not by the author. If the mistakes are all rectified, then reviewer will immediately inform to the test lead.

Test case approval—

This is generally done by the test lead only when he/she comes to know that there are no mistakes present in author’s test cases.

Approval means test case is ready for execution.

Store the test case in test case repository----

Once the test cases are approved by the test lead, author will store all their approved test cases in a centralized location which is called as master test case repository.

What is test case repository---

- A test case repository is a centralized location where all the base line test cases (written, review and approved are stored).
- A test case repository is used to store the approved test cases.
- Any test engineer wants to test the application, then he/she needs to access the test case only from the test case repository.
- For every release, we maintain a different test case repository.
- Once the test cases are stored in the test case repository, they can't be edited or changed without the permission of the test lead.
- The testing team always has a complete backup of the test case repository if any crash happened which is affecting the software.

Test case execution(excel)----

- This is the stage where test engineer checks the functionality of the application (perform black box testing)
- This is the stage where the test engineers check whether the actual result is matching with the expected result or not.
- This is the stage where test engineers will find out the defects in the build.
- This is the stage where test engineers show their productivity to the company.

Test execution report---

- This report is generally prepared by individual test engineers which keys as a track of how many test cases that got passed and how many test cases that got failed.
- This report can be prepared on a daily, weekly, monthly, quarterly basis.
- Test engineers need to share this report to their test lead and finally test lead will combine/merge all the reports and make it as a single test execution report and that report is shared across to---

  I. Project manger
  II. Development team

III. Client/customer.

Total number of test cases present	Total number of test cases executed	Total number of test cases not executed	Total number of test case passed	Total number of test cases failed	% Pass	% Fail
200	60	140	50	10		

Let see one example of test execution report where we have different modules such as sales, amount transfer, tax, loan.

Module name	Total no. of test cases written	Total no. of test cases execution	Total no. of test cases passes	Total no. of test cases fails	Pass %	Fail %
sales	170	170	168	2	98%	2%
Amount transfer	90	90	88	2	97%	3%
tax	127	127	127	0	100%	0%
loan	110	110	109	1	99%	1%
total	497	497	492	5	97%	2%

The test lead made this report, and the test engineer sends the individual feature that he/she has tested and executed.

Test case design techniques----

The test case design techniques that need to be followed by every test engineer while writing the test cases to achieve the maximum test coverage. If we follow the test case design technique, then it become process oriented rather than person oriented.

When ever a tester writes a test case, the main motive/purpose of writing test cases is to catch the defects.

A test is good when our coverage is good. To have a better test case coverage we need to learn the different type of test case design techniques in order to write our test cases.

The different type of test case design techniques are----

I. Error guessing
II. Equivalence partitioning
III. Boundary value analysis
IV. Decision table-based testing
V. State transition technique

I. Error guessing---

it is one of the test case design technique where the test engineer will try to guess the error by entering the negative values. They don't follow any rules. It depends on the analytical thinking of individual test engineer.

Example: ----

Suppose we have one bank account, and we have to deposit some money over there, but the amount will be accepted on a particular range of **which is 5000-7000**. So here, we will provide the different input's value until it covers the maximum test coverage based on the error guessing technique, and see whether it is accepted or give the error message:

value	description
6000	Accept
5555	Accept
4000	Error message
8000	Error message
blank	Error message
100$	Error message

----	----
----	----
Maximum test coverage	

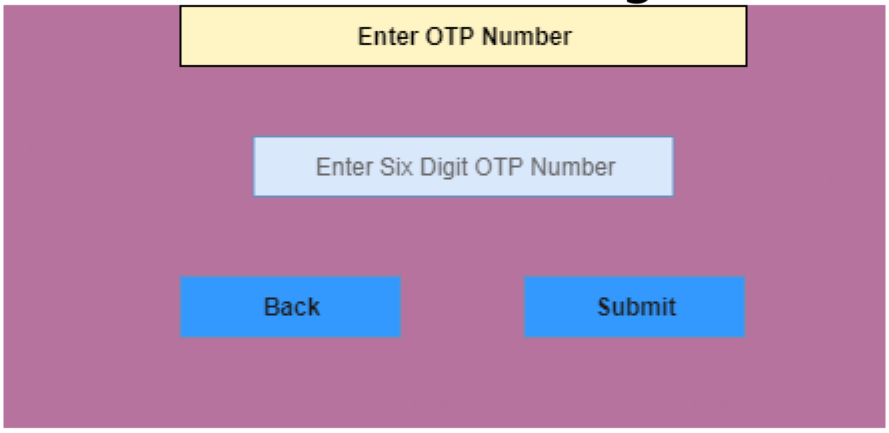
2. equivalence partitioning----

Equivalence partitioning is also called as equivalence class partitioning. In this technique 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. This technique is used to reduce the test cases with best possible test cases which will give good test coverage.

Example: -----

Assume that there is a function of a software application that accepts a particular number of digits, not greater and less than that particular number. For example, an OTP number which contains only six digits, less or more than six digits will not be accepted, and the application will redirect the user to the error page.

OTP Number = 6 digits

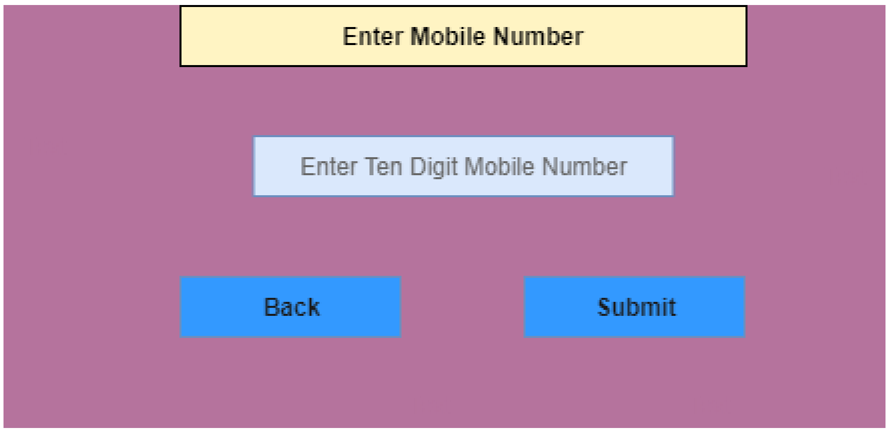


INVALID 1 Test case	INVALID 2 Test case	VALID 3 Test case	VALID
DIGITS >=7	DIGITS<=5	DIGITS = 6	DIGITS = 6
93847262	9845	456234	451483

Example 2: -----

A function of the software application accepts a 10 digit mobile number.

Mobile number = 10 digits



INVALID 1 Test case	**INVALID** 2 Test case	**VALID** 3 Test case	**VALID**
DIGITS >=11	DIGITS<=9	DIGITS = 10	DIGITS =10
93847262219	984543985	9991456234	9893451483

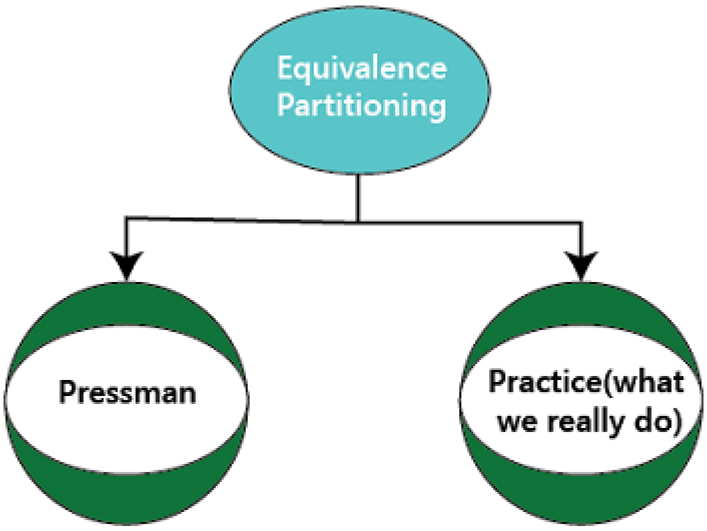
In both examples, we can see that there is a partition of two equally valid and invalid partitions, on applying valid value such as otp of six digits in the first example and mobile number of 10 digits in the second example, both valid partitions behave same, i.e. redirected to the next page.

Another two partitions contain invalid values such as 5 or less than 5 and 7 or more than 7 digits in the first example and 9 or less then 9 and 11 or more than 11 digits in the second example, and on applying these invalid values, both invalid partitions behave same i.e., redirected to the error page.

How we perform equivalence partitioning---

We can perform equivalence partitioning in two ways, which are as follows----

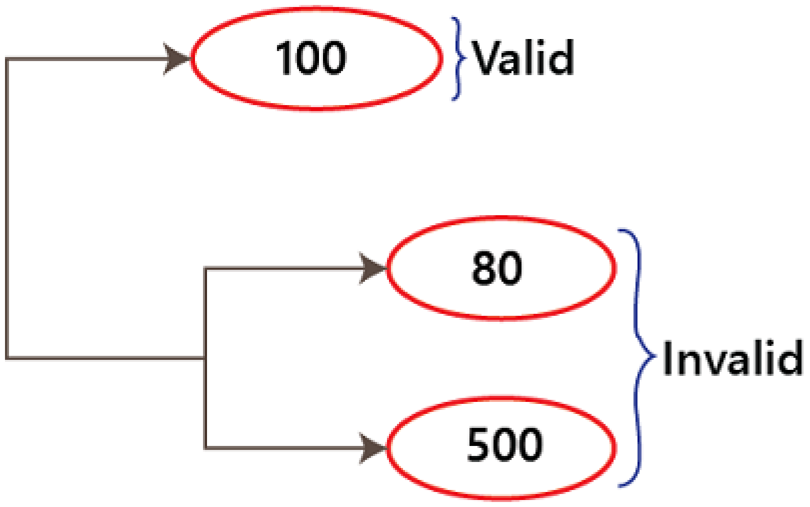
- Press man
- Practice (what we really do)



Lesson1---

When the input is given in the form of rang of values, then design the test cases for one valid and two invalid values.

For example, the Amount of test field accepts a Range (100-400) of values:



When there is a time constraint in our testing project, we can option for lesson1 instead of error guessing.

Lesson2---

Whenever input is given in the form of set of values, then design the test cases for one valid value and two invalid values.

Example-

	Valid	invalid
Monitor---100pc	70	120,0
Mouse-----70pc	20	100,0
Keyboard—50pc	40	200,0

Ecommerce website, select a product---3 units left.

Lesson3---

Whenever input is given in the form of Boolean values then design the test cases for both true as well as false values.

Example---

● **male**		● **female**	
Valid	invalid	invalid	valid
True	false	false	true

Practice---

When the input is given in range of values, then divide the range into equivalent parts and test for all the values and make sure that at least you are going to derive two invalid values and one valid value for each equivalent part.

Example---

Amount **1000---4999**

Valid data

1000—1999 $\rightarrow$ 1500

2000—2999 $\rightarrow$ 2500

3000---3999 $\rightarrow$ 3500

4000--4999 $\rightarrow$ 4500

Ecommerce website, price filter option.

iii. Boundary value Analysis---

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

When ever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

Boundary values are those that contain the upper and lower limit of a variable.

Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be consider as boundary values.

Example

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

```
12  14   15   16  17  18  20 22  24  25 26  28  30 31  32  34  36  38  40
-------------------------------|-------------------------------|-----------------------------
Invalid Partition              Valid Partition              Invalid Partition
```

Invalid test cases	Valid test cases	Invalid test cases
11, 13, 14, 15, 16, 17	18, 19, 24, 27, 28, 30	31, 32, 36, 37, 38, 39

The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error message.

Advantage of boundary value analysis---

- Instead of testing with all the set of input, we can select those data which are part of the boundaries. So, this will help us to reduce the execution time.
- Density of finding out the defects is more at boundaries.

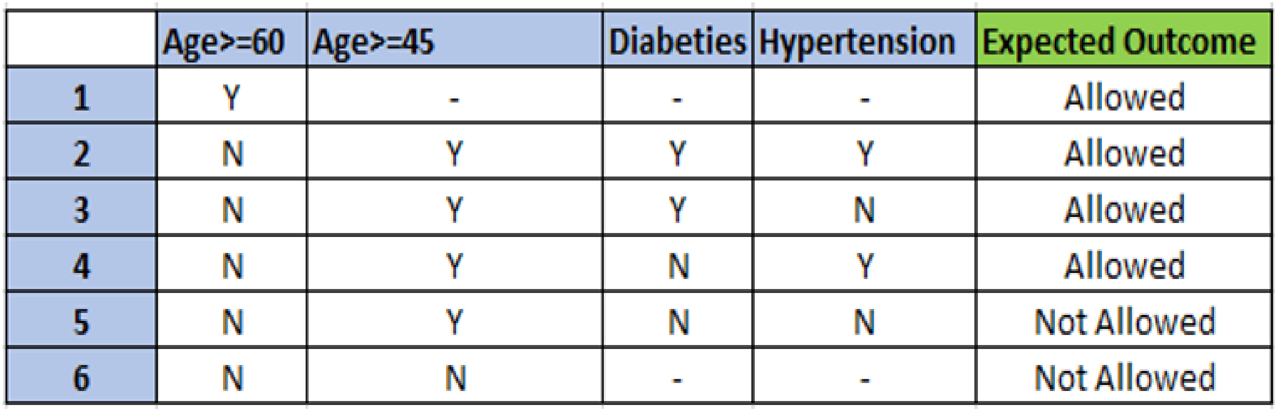
iv. <u>Decision table-based testing-----</u>

A decision table, also known as the cause effect table. This technique is used for functions which respond to a combination of inputs or events.

Example-----build a software that ensure only valid people should get the covid-19 vaccine.

Rule—only people with age>60 years or anyone above 45years with either diabetics or hypertension history should be allowed vaccination.

Now using the decision table technique following should be the test cases.



	Age>=60	Age>=45	Diabeties	Hypertension	Expected Outcome
1	Y	-	-	-	Allowed
2	N	Y	Y	Y	Allowed
3	N	Y	Y	N	Allowed
4	N	Y	N	Y	Allowed
5	N	Y	N	N	Not Allowed
6	N	N	-	-	Not Allowed

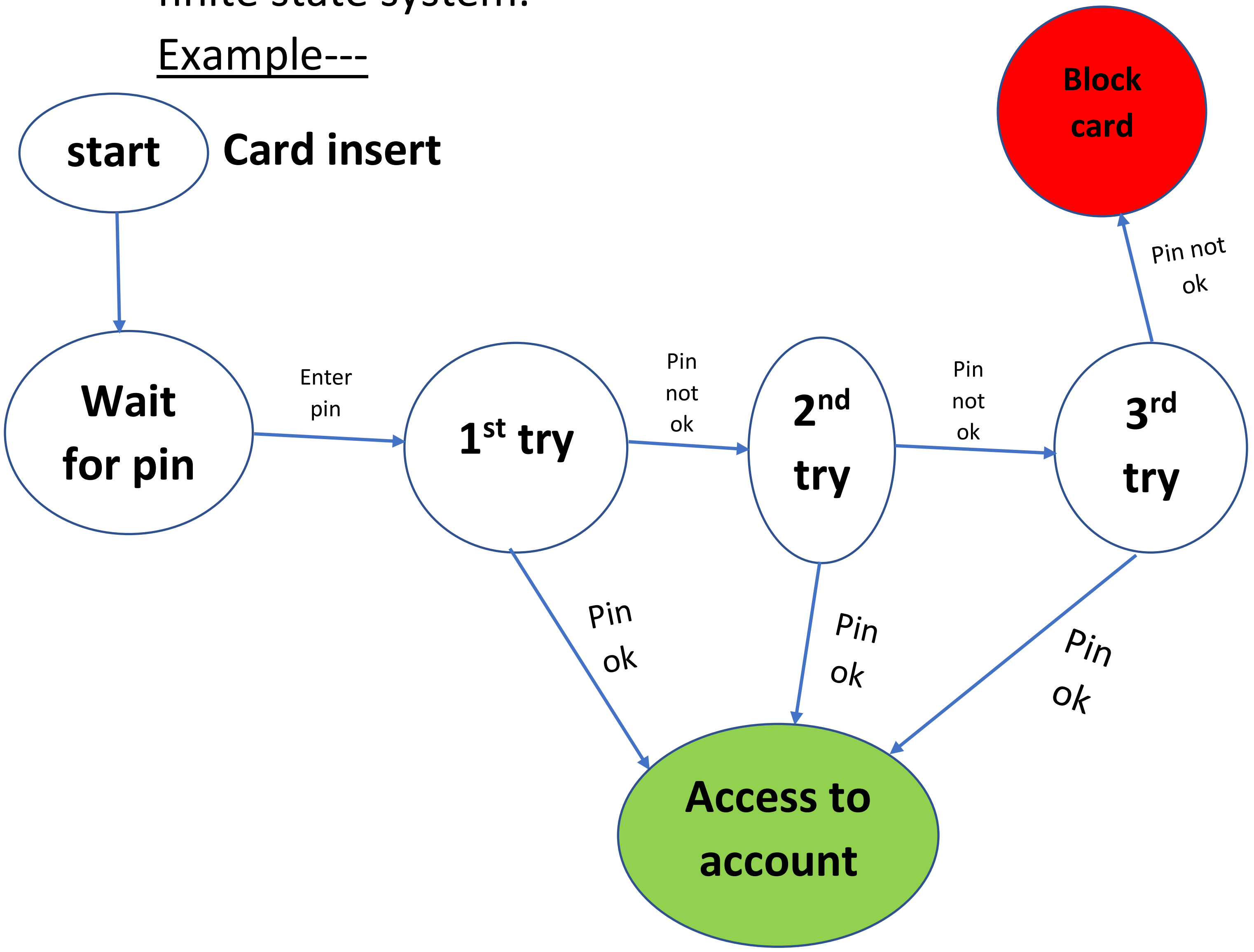
This technique is the easiest way to achieve 100% test coverage.

v. State transition technique----

State transition technique is used when a system has different states and transition from one state to another and determine by the rule of the machine.

Any system where you get different output for the same input depending on what has happen before is a finite state system.

Example---



Here we have 7 states but only 4 possible events. One is card inserted, other one is enter pin, one more is pin ok and one more is pin not ok.

So basically, here we trying inserting our card in atm machine and entering the pin. Once e are entering the pin, if the pin is ok then we are able to access our account or get the money.

If our pin is not ok, after 3 attempts if pin is not ok then our card will be blocked.

So here we are getting different output for the same input (entering pin). It depending on previous state.

Different between defect, bug, error and failure

Defect---

When the application is not working as per the requirement then it is known as defect. The variation between the actual result and expected result is known as defect.

In other word, we can say that the bug announced by the programmer inside the code is called a defect.

Bug---

It is an informal name given to any defect. This term is generally used by the testing team.

Error---

It is the mistake occur due to the developer’s coding. Because of this error programmer/developer not able to compile or run the program. This term is generally used by the development team.

Failure----

If there are defects/bugs/errors in an application, the application will lead to a failure (stop working).

One single defect can cause a failure or multi defects can cause a failure to an application.

In other word we can say that if an end user detects an issue in the product, then that particular issue is called a failure.

For example—

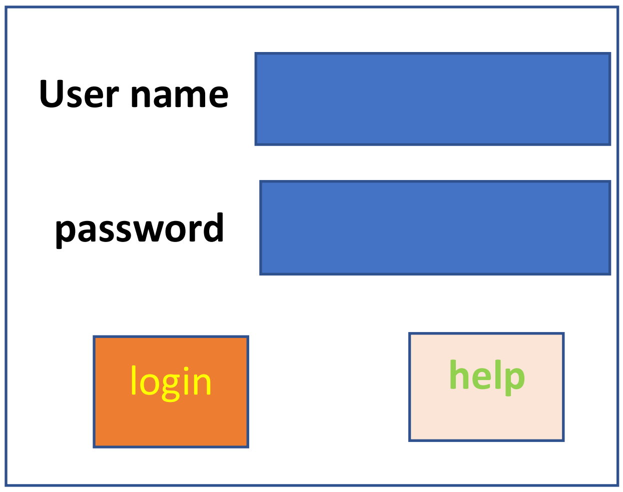
In a bank application if the amount transfer module is not working for end users. when the end user tries to transfer money, submit button is not working, hence this is a failure.

<u>Why we get defect/bug in a software? ----</u>

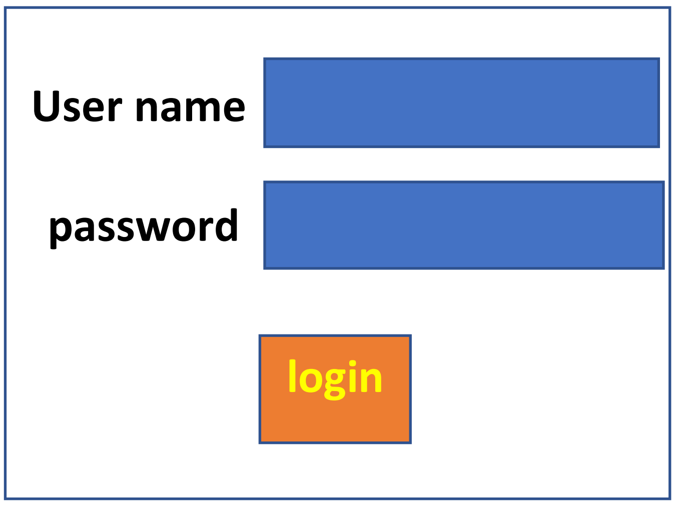
i. **<u>Wrong implementation---</u>**
   Here wrong implementations means coding. For example-in an application when you click on sales link, it goes to purchase page. This occurs because of wrong coding. Thus, this is a bug.

ii. **<u>Missing implementation---</u>**
We may not have developed the code only for that feature.

Example---



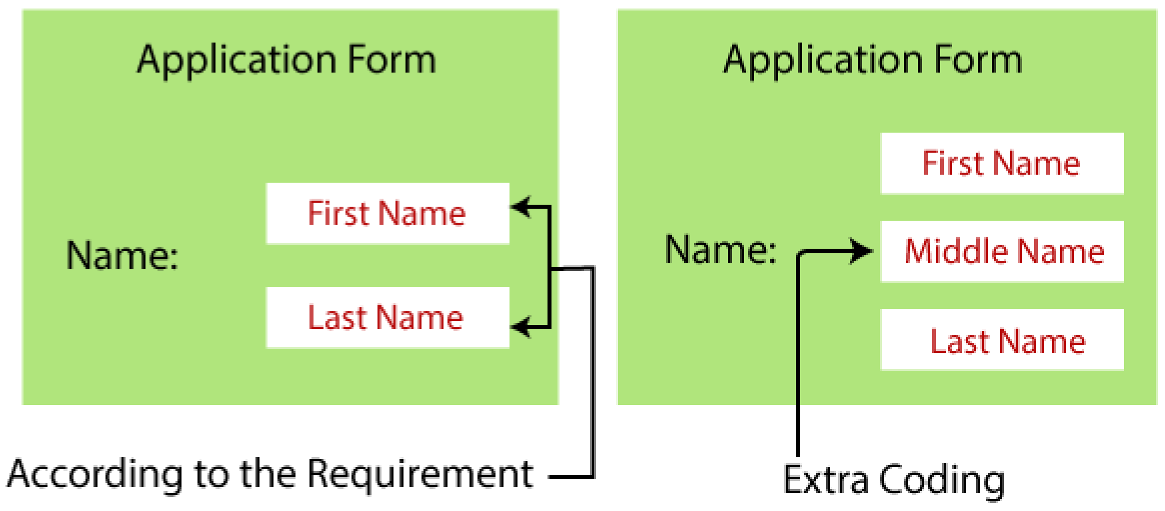
Customer's requirement



Developed by the developer

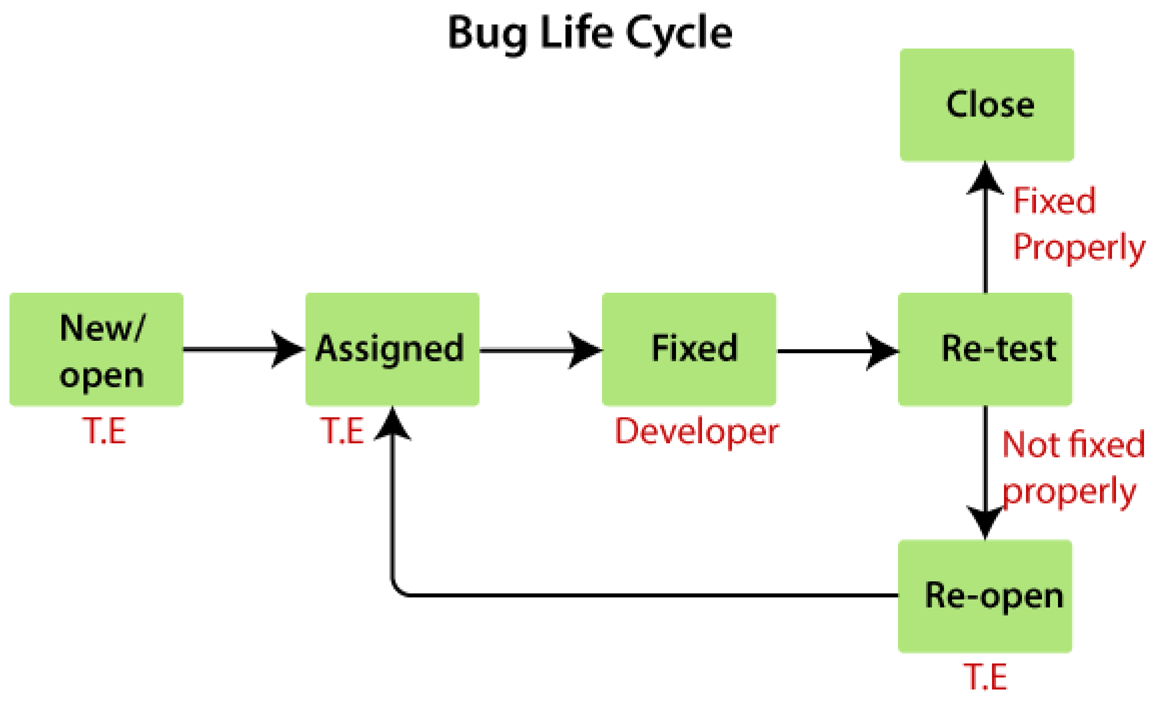
(Help option is missing)

iii. **<u>Because of extra implementation----</u>**
It means something is not requested by the customer and it is developed by the developer.



Defect life cycle/bug life cycle----

Bug life cycle is also known as defect life cycle. It is a process in which defect goes through different stages in its entire life. This life cycle starts as soon as a bug is reported by the tester and ends when a tester ensure that the issue is fixed and won't occur again.



As soon as test engineer finds a defect, he prepares a defect report set the status as open and sends it to development lead and put cc (carbon copy) to test lead.

As soon as development lead gets the defect report, he will go through the defect report and he will easily come to know which developer has done the mistake and which testing team has found the defect. He will then change the defect report status to assigned and send it to the developer and put cc to test engineer.

Developer as soon as he receives the defect report, will go through the report and will easily come to know where exactly he has done the mistake. So he will go to the source code of the application and he will fix the defect. Fixing the defect is nothing but modifying the program. After fixing the defect, he will change the defect report status to fixed and he will send a mail to the test engineer and put cc to development lead.

Test engineer after receiving the mail from the developer will come to know that the defect has been fixed. Then, he will do testing to check whether the earlier defect are fixed or not. If all the defects are fixed, then he will change the defect report status to closed. If the defects are still there, then he will change the defect report status to reopen and again send it to the developer. So this process goes on until the defects are fixed.

Why do test engineer put cc (carbon copy) to test lead? –

Because, the test lead is the person who keeps on attending meeting with the development team and customer. So, he should be aware of what exactly is happening in the project and to get the working visibility of the test engineers**.**

What is age of the defect?

The time duration taken from identifying the defect till fixing and closing the defect.

Reject status: ---

Test engineer finds a defect, prepare a defect report and send it to the development team, but they are saying that it is not a defect (invalid data), that is development team are not accepting it as a defect.

Why we get defect report status as reject?

- **Because of misunderstanding the requirement of the customer.**
  Example—
  User name

Customer requirement—it should accept a combination of 1-10 characters and numerals.

Test engineer--- it should accept 1—10 characters only.

<u>Invalid data—</u>

Kuresh123 (test engineer expecting an error message)

But in real time data will accept. Test engineer prepare defect report and send it to the development team---- reject.

- **Because of referring the old requirement of the customer----**
  Example---
  User name
  Customer's requirement (old)--- it should accept a combination of 1—10 characters and numerals.
  Test engineer----it should accept a combination of 1—10characters and numerals.
  Customer's requirement (new)---- it should accept a combination of 1-15 characters and numerals.
  <u>Invalid data---</u>
  12charnum (test engineer is expecting an error message).
  But real time the data will accept.
- **Because of improper installation of the software.**

Postponed/deferred status---

Test engineer finds a defect, prepares defect report and sends it to the development team but they are saying that they can't fix the defect immediately rather they will fix it later.

Why we get defect report status as postponed/deferred?

When ever developers are busy in fixing the critical defects of the application, same time parallelly test engineer is busy in sending minor defects to the development team.

- **Duplicate status of the bug—**
  When the same bug has been reported multiple times by the different test engineers are known as a duplicate bug.
  Why we get defect report status as duplicate---
  i. Because of common feature assigned to multiple test engineers.
     For example---
     Suppose we have test engineer P and Q which are testing the software. The test engineer P and Q will test their features like login the application. Here the test engineer P enters the valid username and password and click on the login button.

Once P click on the login button, it opens a blank page, which means that it is a bug. After that P prepares a bug report for the particular bug and sends it to the developer.

Then the test engineer Q also login the application and got the same bugs. Q also prepare a bug report and send it to the developer.

Once the developer got both test engineers bug report, he/she sends back the bug report to the Q and say it is duplicate.

ii. Dependent module.

How to avoid duplicate defect report status?

Before sending any defect reports to the development team, the test engineer should first go and check in the defect repository whether the defect has been already logged by some other test engineer or not.

If the defect is already logged, then the test engineer should forget about that defect and start identifying some other defects. But if the defect is not present inside the defect repository, then test engineer should prepare a defect report and send the same to the development lead and keep one copy of the defect report inside the defect repository.

<u>**Cannot be fixed status---**</u>

Test engineer finds a defect, prepare defect report and sends it to the development team but they are saying that they can’t fix the defect.

Cannot be fixed status is generally given to minor defects.

Why we get defect report status as cannot be fixed: -

Following are the reasons for the can’t fix bug/defect.

- <u>No technology support---</u>
  The programming language used by the developer to develop the software not having capacity to fix the defect.
    - If we need some third-party tools then it will more investment.
    - We need skilled resources—more investment.
- If the defect is identified in the root of the product/application.

  If the bug is minor (not important and does not affect the application), the development lead says it can be fixed in the next release.
- If the cost of fixing the defect is more than cost of defect.

Investment that will be done by the software organization to fix all the minor defects (1000 dollar)

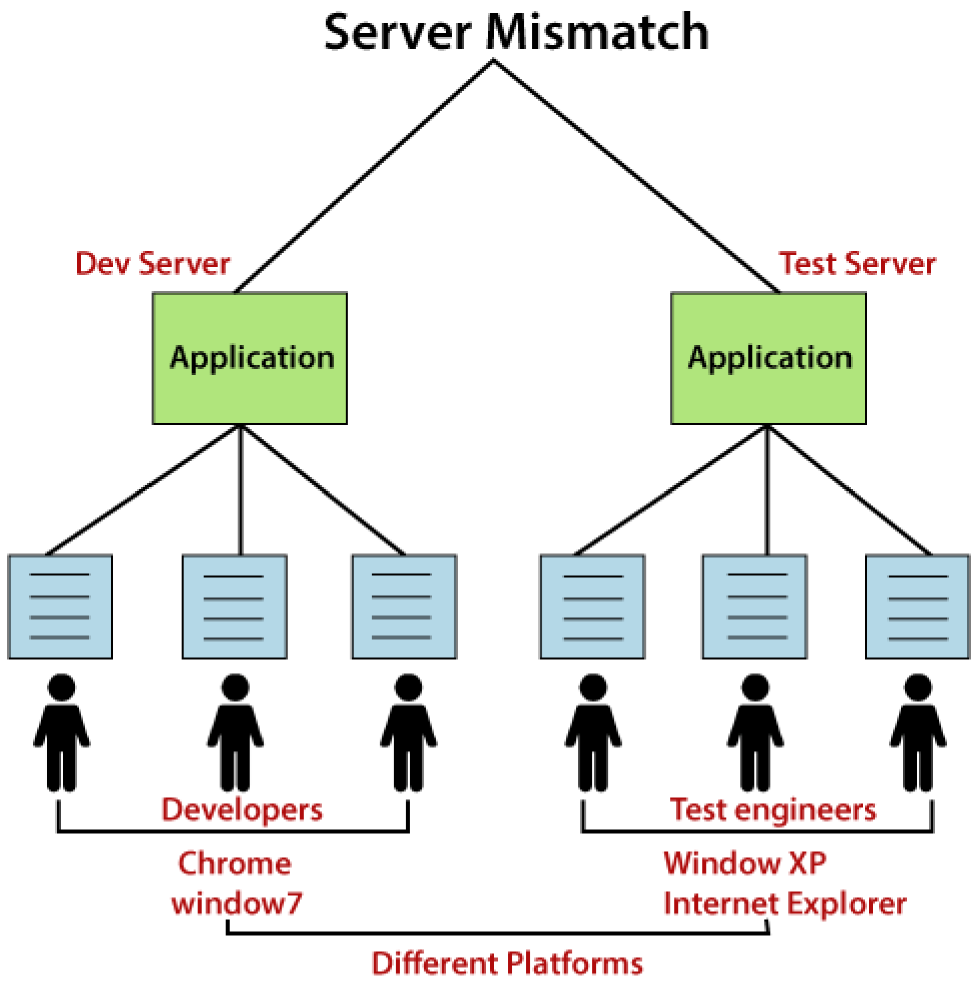
Impact of the defect on the business of the customer/client (100 dollar).

<u>Not reproducible status---</u>

Test engineer finds a defect, prepares a defect report and sends it to the development team but they are saying that they are not able to reproduce of find the defect.

Why we get defect report status as not reproduceable---

- Because of environment mismatch (server and platform mismatch)---



- <u>Because of build mismatch-</u>

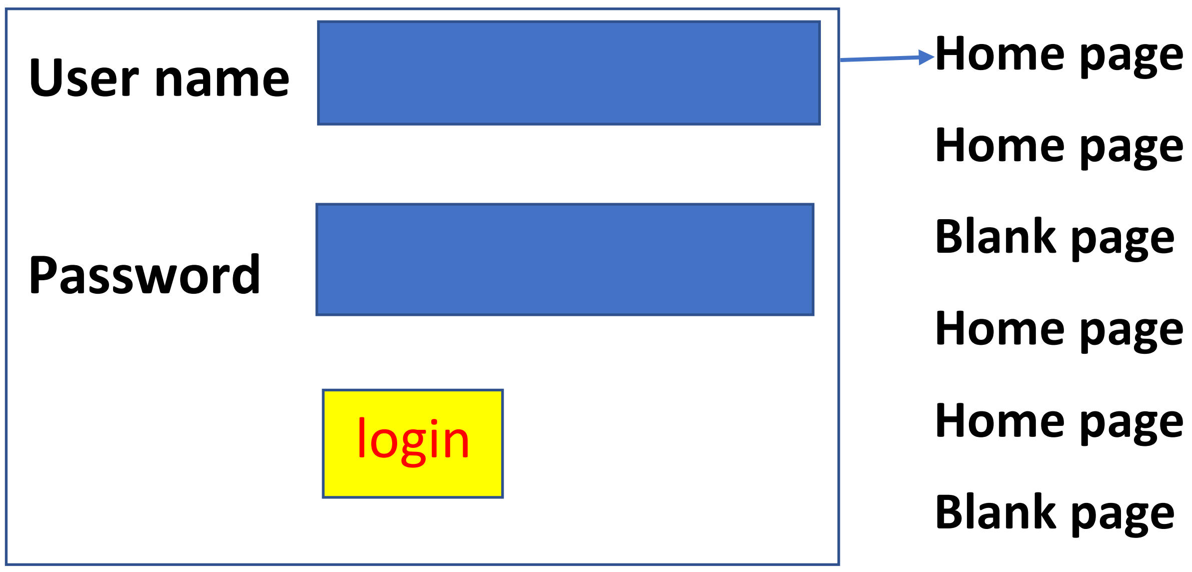
  the test engineer will find the bug in one build, and the developer is reproducing the same bug in another build.

<u>Test engineer</u>	<u>developer</u>
Build no-10	build no—9

- <u>Because of inconsistent defect/bug---</u>

  The bug is found/coming at some time, and some time bug is not coming.

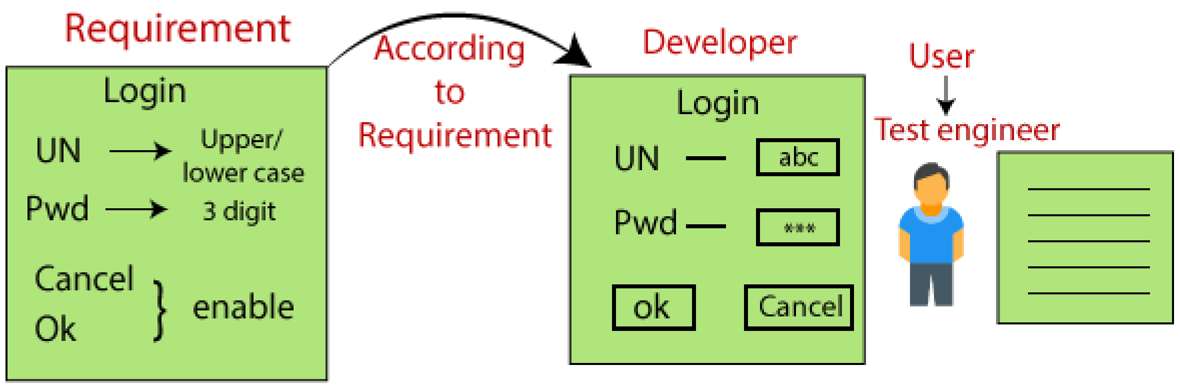
Example---



whenever we enter valid user name and valid password and click on login button it is going to a blank page multiple time.

- Because of improper defect report preparation---
  The test engineer did not mention the complete navigation steps in the report.
  Request for enhancement (REF)---(any new feature getting added to the software)
  These are the suggestions given by the test engineer towards the enhancement of the application in the form of a bug report.
  The test engineer thinks that the look and feel of the application or software are not good because the test engineer is testing the application as an end user, and he/she will change the status as RFE. If the customer

says yes, then the status should be fix. If the customer says no, then the status should be close.



Sir’s note---
Test engineer find a bug and sends it to development team. When development team look at the report send by the test engineer, they know it is a bug. But they say it is not a bug because it is not part of the requirement.

Defect report/bug report

A bug report in software testing is a detail document about bugs found in the software application. Bug report contains each detail about bugs like description, date when bug was found, name of the tester who found it, name of the developer who fixed it etc. Bug report helps to identify similar bugs in future, so it can be avoided.

Bug report is generally prepared by the test engineer as soon as he/she find a defect or bug in the build while performing testing. Once the defect report is prepared, test engineer needs to share that report to the development.

For every defect/bug, separate defect report should be prepared.

If the actual result coming in the application is not matching with the expected result, we consider that as a defect.

Defect report can be prepared using—

I. Excel sheet
II. Google sheet
III. Defect tracking tool like Jira, mantis, Bugzilla.

Example---

Let’s assume test engineer composed on mail and the mail is not getting displayed in sent mail page.

A sample defect report template------

Defect id	d-009
Module name	Sent mails
Test case name	Gmail-sent mail
Build no	b—10
Test environment	Windows 10, chrome
Status	New/assigned/fixed/reopen/reject
Severity	Blocker/critical/major/minor
Priority	High/medium/low
Expected result	Mail should be displayed in sent mail page.
Actual result	Mails are not displayed in sent mail page.
Detail description	I. Open the browser and enter the URL. II. Click on compose button III. Enter valid data in all fields and click on send button. IV. Click on sent mail link.
Found by	Test engineer’s name (kuresh)

The defect report varies from company to company. But the following are the mandatory attributes of a defect report in all companies. These are---

I. Defect id
II. Severity
III. Priority

i. **Defect id---**

Defect id is a serial number of defects in the report.

ii. **Severity (impact)---**

Severity is the impact of the defect on the business of the customer.

To define severity, we use the following terms---

a) Blocker
b) Critical
c) Major
d) Minor

a) <u>Blocker---</u>

These kinds of defects will completely block the business of the customer. These kinds of defects will have lots of impact on the business of the customer.

Example—

User is unbale to create an account. User is unable to login to the account. Whenever user enters valid username and valid password, blank page is displayed.

b) <u>Critical defect---</u>

These kinds of defects will also have lots of impact on the business of the customer. If we get any critical defects, we should stop the testing.

Example---

Compose feature is not working.

Inbox feature is not working

Logout feature is not working.

c) <u>Major defect---</u>

These kinds of defects will have little impact on the business of the customer. If we find these defects, no need to stop the testing.

Example—trash feature is not working.

d) <u>Minor defects---</u>

These kinds of defects will have no impact on the business of the customer.

<u>Example---</u>help feature is not working, spelling mistakes is minor features.

Who sets the severity and priority of the defect?

Test engineer sets the severity and priority of the defect. But priority can be changed by the developers**.**

iii. <u>Priority----</u>

It is the importance given to fix the defect, that is which defect has to be fixed first and which defect has to be fixed at the end.

to define priority, we use the following terms: ---

- High or p1
- Medium or p2
- Low or p3

Last before release/deployment, you got a critical defect in the application. Will you launch the product to the customer?

As a test engineer, we are not decision taking bodies in the organization. Our role is just to find out the defect, prepare defect report and send it to the development team.

We can suggest our manager that if we launch the product with that critical defect, it might affect the business of the customer as well as the software organization.

As test engineer you found a critical defect. In the build, you prepared a defect report and send it to the development team. But they are rejecting your defect again and again. What will be your approach as a test engineer to convince the developer?

- Take a screenshot of the defect.
- Take a video of the defect.
- Share the screen and give a demo of the defect and try to make the developer understand the impact of the defect on the customer's business.
- Communicate with our test lead.

Defect reporting process: ---

- **For small scale company----**

$$\begin{tikzcd}\text{Test engineer} \arrow[d] \\ \text{Developer}\end{tikzcd}$$

- **For medium scale company---**

$$\begin{tikzcd}\text{Test engineer} \arrow[d] \\ \text{Development lead} \arrow[d] \\ \text{Developer}\end{tikzcd}$$

- **For large scale company---**

$$\begin{tikzcd}\text{Test engineer} \arrow[d] \\ \text{Test lead}\end{tikzcd}$$

$$\begin{tikzcd} {} \arrow[d] \\ \text{Development lead} \arrow[d] \\ \text{Developer} \end{tikzcd}$$

What is defect masking?

Masked defect is a defect that hides other defects in the system.

Example---

There is a link to add employee in the system. On clicking this link, you can also add a task for employee. Let's assume, both the functionalities have bugs. However, the first bug (add an employee) goes unnoticed. Because of this the bug in the add task is masked.

Why we get new defects/bugs in old features?

I. Adding new features/modifying existing features might affect the old features and create new defects.
II. Fixing one bug in old feature might have an impact on some other older features.
III. Chances are there we might have missed that defect in the previous test cycle.

How will you make sure that your test coverage is good/100%?

I. I will invest a lot of time in reading and understanding the requirement document.
II. While writing the test cases, I will apply test case design techniques.
III. We can conduct a brain storming session.
IV. Once the test cases are written, we generally review the test cases.
V. I will conduct adhoc testing to cover some more scenarios.

Types of application

i. Standalone application /desktop application
ii. Web application
iii. Client server application
iv. Mobile application
v. Gaming application

i. **<u>Standalone application—</u>**

Standalone application is an application that runs locally on the device and does not require anything else to be functional.it does not need an internet connection nor any other services installed.

Standalone application can be developed by using programing languages. Standalone applications are installed on a personal or work computer.

Example---notepad, calculator, Microsoft ward.

Standalone applications are faster in access. The stored data in it are secured from data hacking and virus. Only single user can access it at a time.

ii. **<u>Web application---</u>**

Web applications are not needed to install on the system. We can directly access it by using web browsers through internet. These application can developed only by using web related technologies.

Example--- Gmail, YouTube, Facebook. Uses of web application is totally dependent on internet connectivity and speed.

Absence of internet or its poor connectivity can cause performance issue with web applications.

We can access the web applications from any location using the internet through the world.

iii. **Client server application---**

An application that run on the client side and access the remote server for information is called a client server application. The client server always makes request to the remote server to get some information.

Example--- skype desktop, team viewer, atm machines.

iv. **Mobile application---**

A mobile application is a type of application designed to run on a mobile device, such as smart phone or tablet.

A mobile application also may be known as an app, web app, online app or smartphone app.

Apps are generally small, individual software units with limited function.

Example---WhatsApp, messenger, Instagram etc.

v. **Gaming application---**

The application which is developed for entertainment purpose.

Example—free fire, cod, bgmi, coc etc.

Different between quality assurance (qa) and quality control (qc)---

Quality—

Quality is meeting requirement, expectation and needs of the customer is free from the defects. There are standards needs to follow to satisfy the customer requirements**.**

Assurance---

Assurance is provided by the organization management. It gives a security that the software will work without any glitches as per the expectation of the client/customer.

Quality Assurance---

Quality assurance is known as QA and focus on preventing defect. Quality assurance has to complete before quality control.

Control---

Control is to test or verify whether the actual result is matching with the expected result or not.

Quality control----

Quality control is known as QC and focuses on identifying a defect. Quality control has to complete after quality assurance.

	Quality assurance		**Quality control**
I.	Qa is a process that aims at preventing the defects in the initial stage.	I.	Qc is a process that aims at identifying and rectifying the defects.
II.	Qa is a technique which is used to manage the quality(verification).	II.	Qc is a technique which is used to verify the quality(validation).
III.	Qa does not involve execution of the programs or execution of the codes.	III.	Qc involve execution of the programs or execution of the codes in order to check the functionality.
IV.	Qa is consider as a preventive measure.	IV.	Qc is considered as a corrective measure.
V.	Qa is responsible for full software development life cycle.	V.	Qc is responsible for software testing life cycle.
VI.	Qa requires both development and testing team.	VI.	Qc requires only the testing team.

Different between static testing and dynamic testing-

Static testing is a testing, which checks the application without executing the code. It is a verification process. Some of the essential activities are done under static testing such as business requirement review, design review and the test documentation review.

Static testing is performed in the white box testing phase, where the programmer checks every line of the code before handling over to the test engineer.

Static testing can be done manually or with the help of tools to improve the quality of the application by finding the error at the early stage of development, that why it is also called the verification process.

Dynamic testing is a testing, which is done when the code is executed at the run time environment. It is a validation process where functional testing (unit, integration and system testing) and nonfunctional testing (user acceptance testing) are performed.

We will perform the dynamic testing to check whether the application or software is working fine during and after the installation of the application without any error.

Static testing

I. **Static testing is performed in the early stage of the software development.**
II. **In static testing whole code is not executed.**
III. **Static testing prevents the defect.**
IV. **Static testing is less costly.**
V. **It generally takes short time.**
VI. **Static testing is performed before code deployment.**

Dynamic testing

I. **It is performed at the later stage of software development.**
II. **In dynamic testing whole code is executed.**
III. **Dynamic testing finds and fixes the defect.**
IV. **Dynamic testing is highly costly.**
V. **It is usually takes longer time as it involves running several test cases.**
VI. **Dynamic testing is performed after code deployment.**

difference between verification and validation

verification---

verification is the process of checking that a software achieves its goal without any bugs.

It is the process to ensure whether the product that is developed is right or not. It verifies whether the development product fulfills the requirements that we have.

Verification is a static testing. Verification means are we building the product, right?

Validation----

It is the process of checking the validation of product. i.e., it checks what we are developing is the right product. Validation is the dynamic testing.

Validation means are we building the right product?

Verification	Validation
I. It includes checking documents, design, codes and programs.	I. It includes testing and validating the actual product.
II. Verification is the static testing.	II. Validation is the dynamic testing.
III. It does not include the execution of the code	III. It includes the execution of the code.
IV. Verification uses method like walk through, inspection and review.	IV. Validation uses methods like functional and non functional testing.
V. It identifies bugs in the early stage of product development.	V. It identifies bugs that are not caught in the verification stage.
VI. It comes before validation.	VI. It comes after verification.

Stub and driver: -----

In software testing, the words stub and driver describe as a replica/dummy of the modules that operate as an alternative to the new module or modules which are in their developing stage, missing or not developed yet.

Stubs are mainly used in the top-down integration testing. On the other hand, drivers are mainly used in bottom-up integration testing.

Generally, stubs are created by software developer.

Example—

Suppose you are told to test a website whose corresponding primary modules are, where each of them is interdependent on each other, as follow—

Module A—login page website

Module B—home page of the website.

Module C—profile setting.

Module D—sign out page.

Assume module A is developed. As soon as it is developed, it under goes testing, but it requires module—B, which is not developed yet. So in this case, we can use the stubs or drivers that simulate all features and functionality that might be shown by actual module—B. so we can say that stubs and drivers are used to fulfill the necessity of unavailable modules.

Similarly, we may also use stubs or drivers in place of module C and module D if they are too not available.

Do both drivers and stubs serve the same functionality?

Yes, we can say both serve the same feature and are used in the absence of a module that has interdependencies with on other module, that is need to be test. So we use drivers or stubs in order to fulfill module's unavailability's and to serve its functionality.

Different between stubs and drivers: ---

Stubs

I. Stubs are used in top down integration testing.
II. Stubs are basically used in the unavailability of low level module.
III. Stubs are taken into used to test the feature and functionality of the module.
IV. The stubes are taken into Concern if testing of upper levels of the modules are done and the lower levels of the modules are under developing process.
V. Stubs are used when lower level of modules are missing or in a partially developed phase, and we want to test the main module.

Drivers

I. Drivers are used in bottom up integration testing.
II. While drivers are mainly used in place of high level modules and in some situation as well as for low level modules.
III. Where as drivers are used if the main module of the software is not developed for testing.
IV. The drivers are taken into concern if testing of lower levels of the modules are done and the upper levels of the modules are in developing process.
V. Drivers are used when higher level of modules are missing or in a partially developed phase, and we want to test the lower module(sub module).