Testing:

Something is there and we are checking whether it is working or not according to the client requirement.

OR

Testing is the process of verification and validation of an application.

In **verification**, we check whether the application is properly implemented or not.

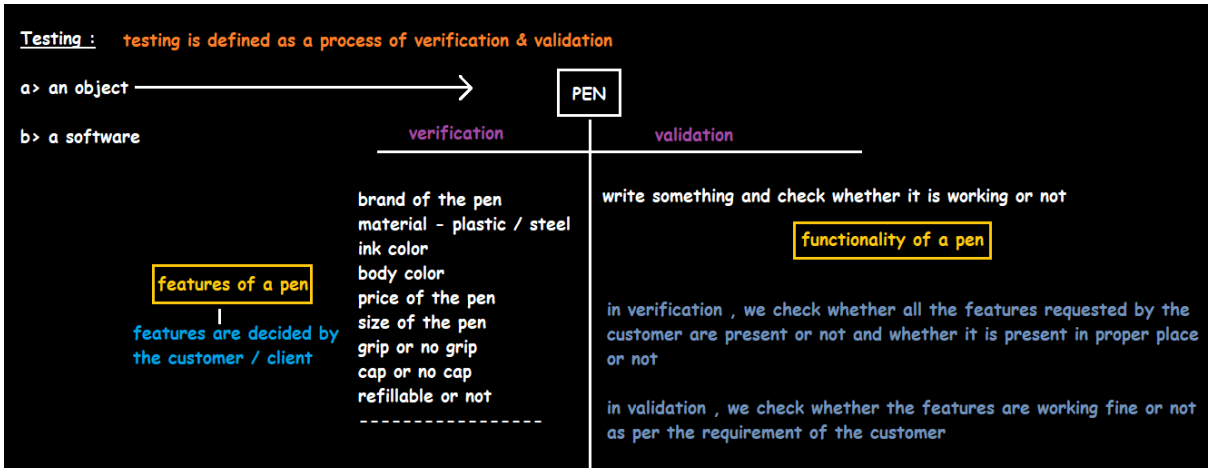
For eg, marker is there, we are checking whether cap, name and price of the marker is there or not.

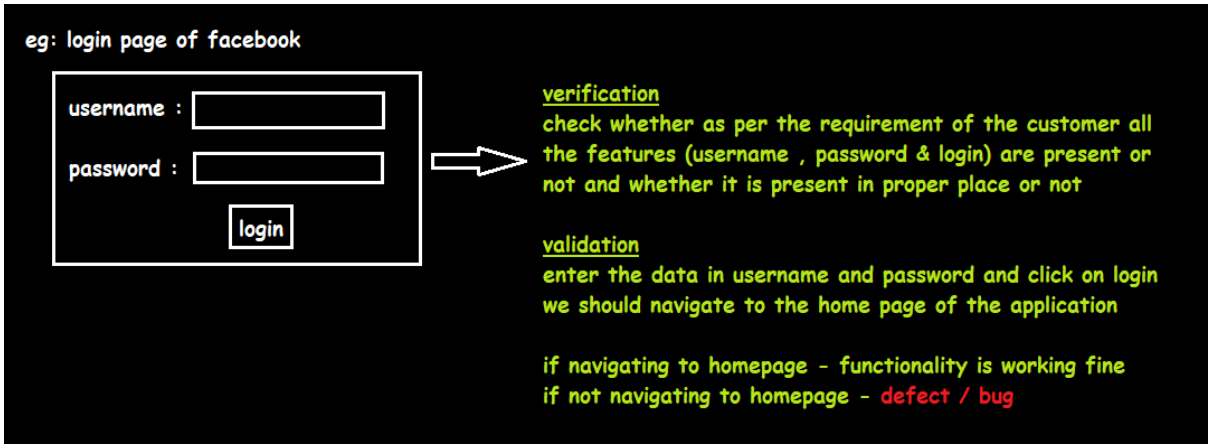
It is a kind of static testing. Means, we are trying to find out the mistakes/defects/bugs/error without executing the application.

In **validation**, we are checking whether the application is functionally working or not.

It is a kind of dynamic testing. Means at the time of execution, we identify the mistakes/defects/bugs/error.

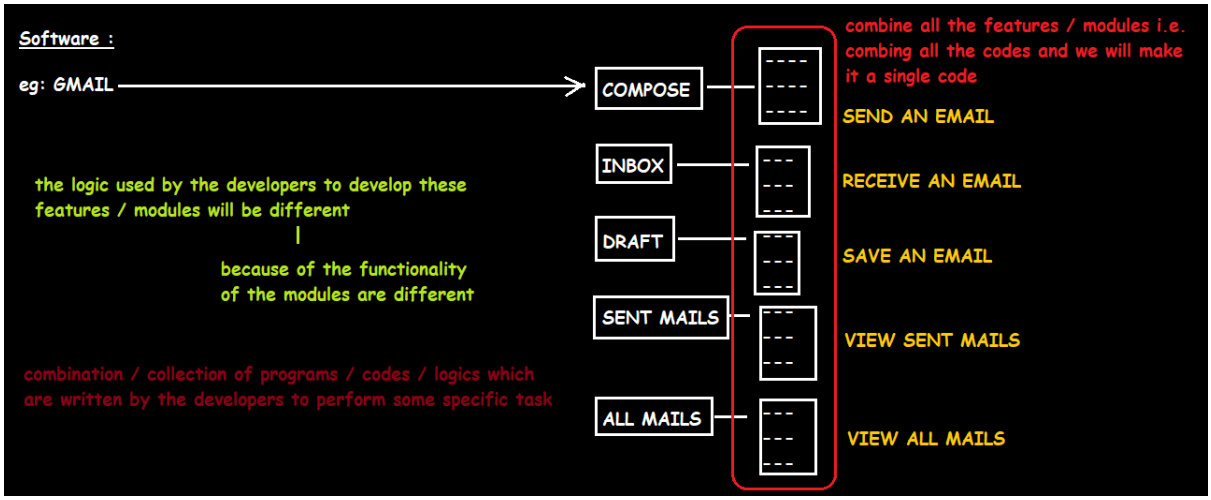
For eg, marker is there, we are checking whether we are able to write or not using the marker. That is nothing but we are checking the functionality of the marker.





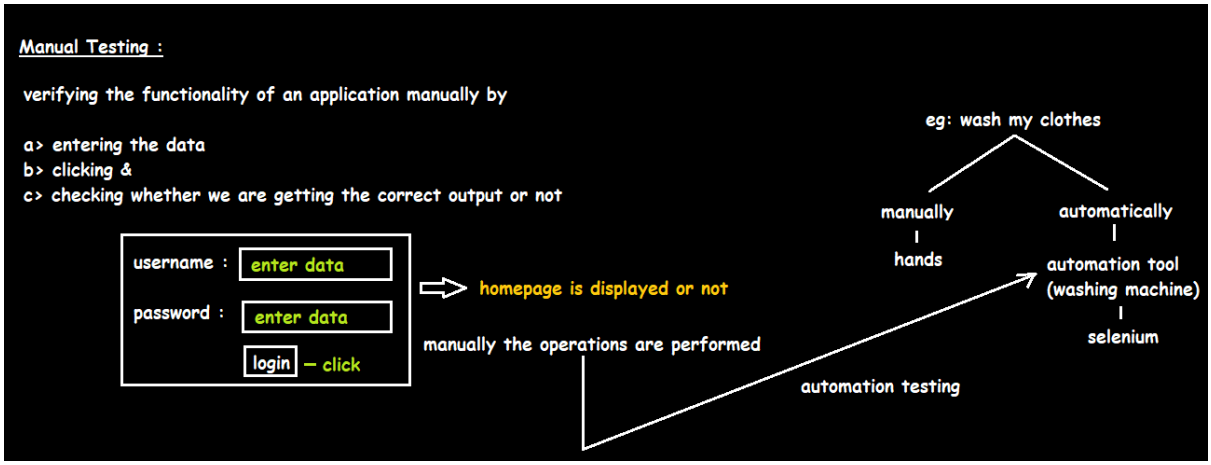
Software:

Combination of programs/codes/logics which are written by the developers to perform some specific task.



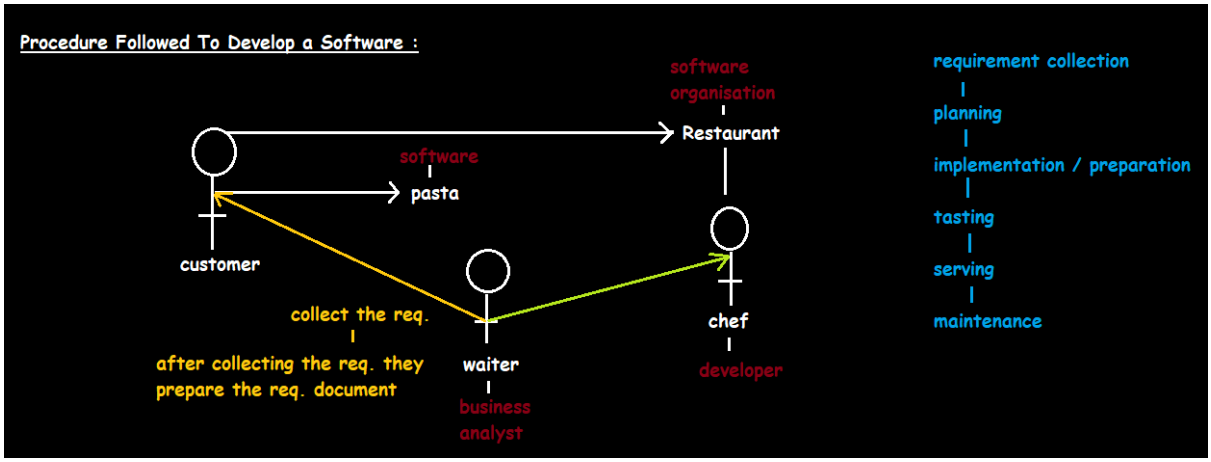
Manual Testing

Verifying the functionality of an application manually by entering the data , clicking & checking whether we are getting the proper output or not.



How to develop a software ?

Developers directly don't start with the writing of the programs. They follow a well defined cycle to develop a software and that cycle is called as **SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)**.



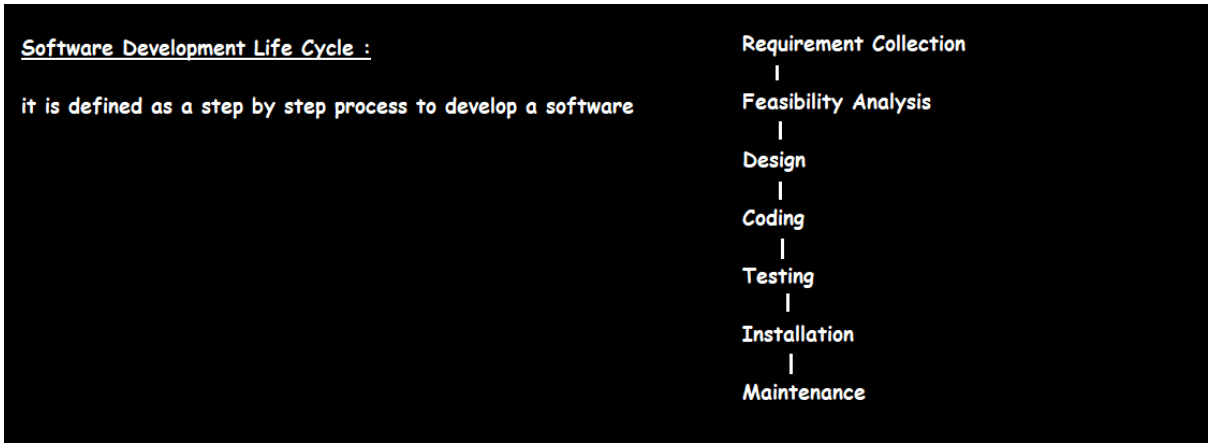
Software Development Life Cycle (SDLC):

It is a step by step procedure to develop the software.

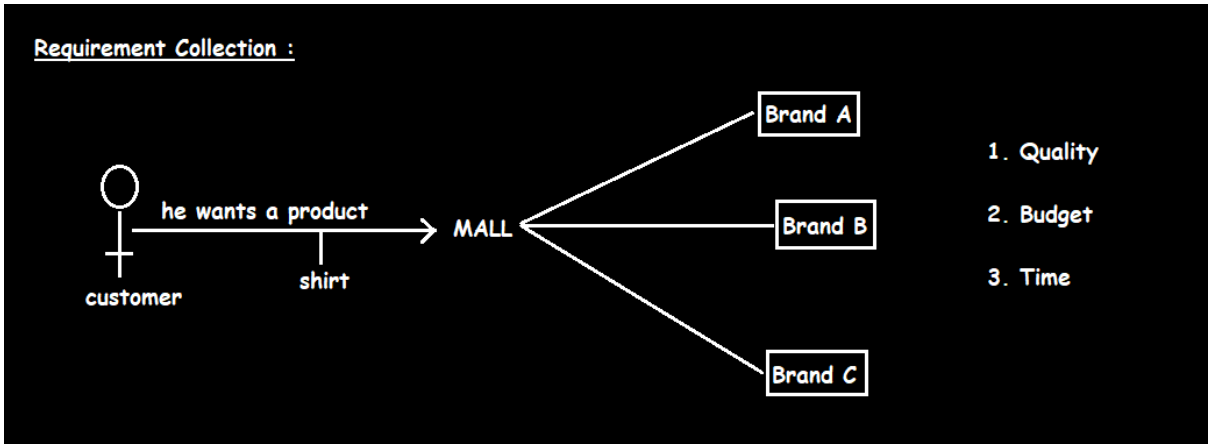
Any SDLC should result in a high quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently and is inexpensive to maintain and cost effective to enhance.

Stages of SDLC:

1. Requirement Collection
2. Feasibility Study / Analysis
3. Design
4. Coding
5. Testing
6. Installation
7. Maintenance



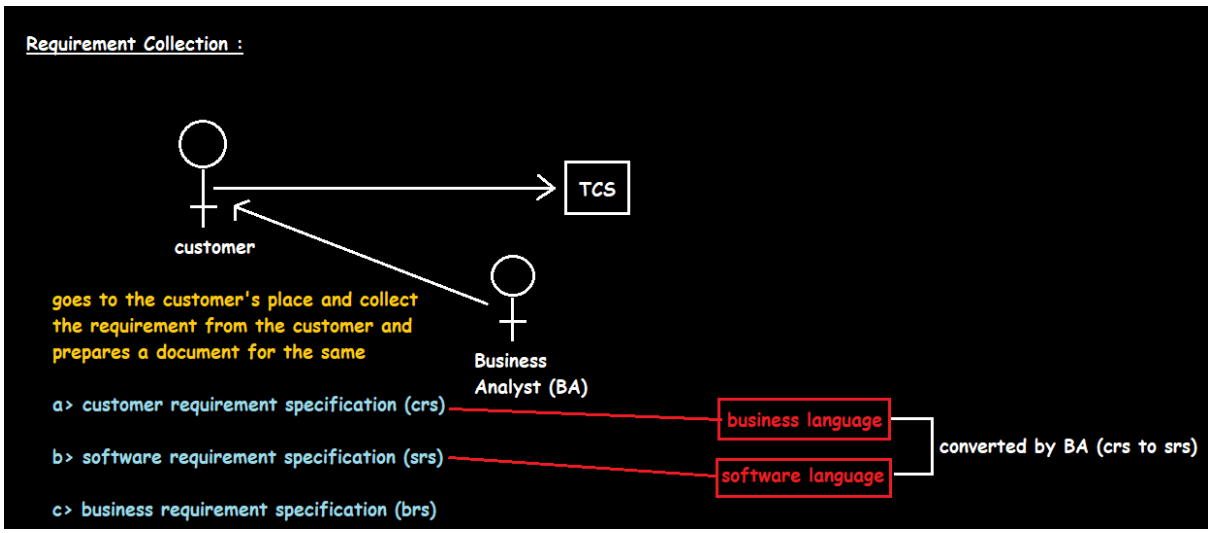
Requirement Collection:

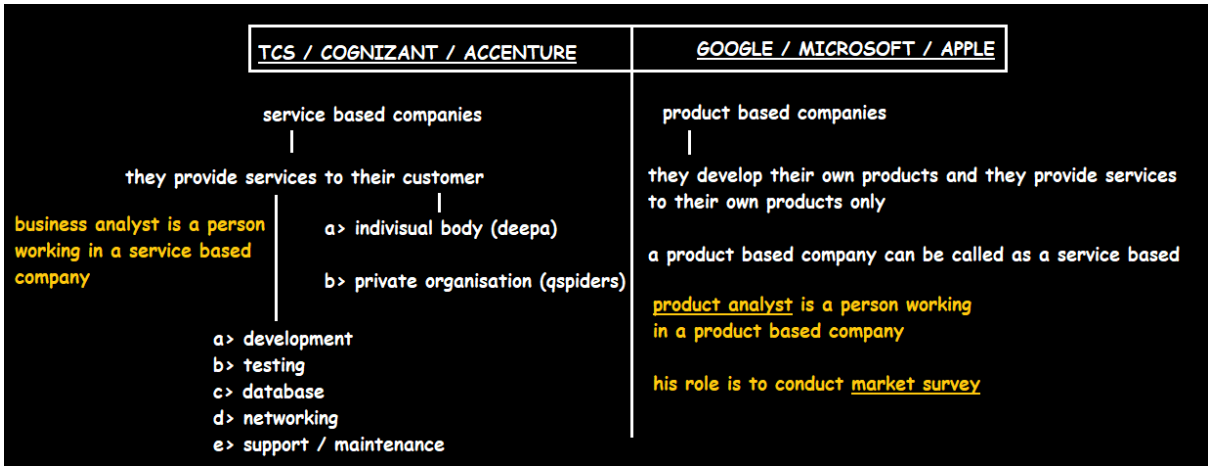


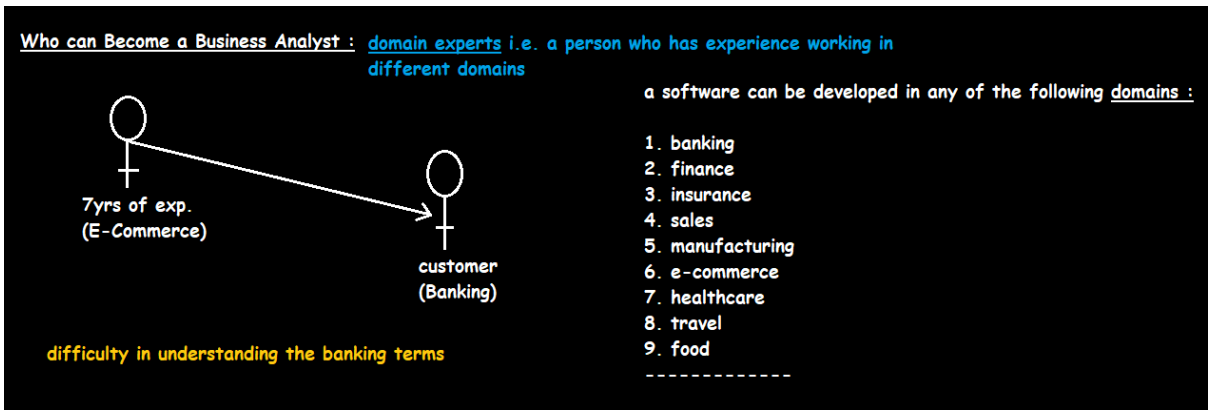
- done by **Business Analysts** wherein he goes to customer's place and collect the requirement and immediately prepares a document for the same.

- gathering requirements in the form of Software requirement specification (**SRS**)/Business Requirement Specification (**BRS**)/ Software requirement Document (**SRD**)/ Business Requirement Document (**BRD**)/ User Requirement Specification (**URS**)/ User Requirement Document (**URD**).

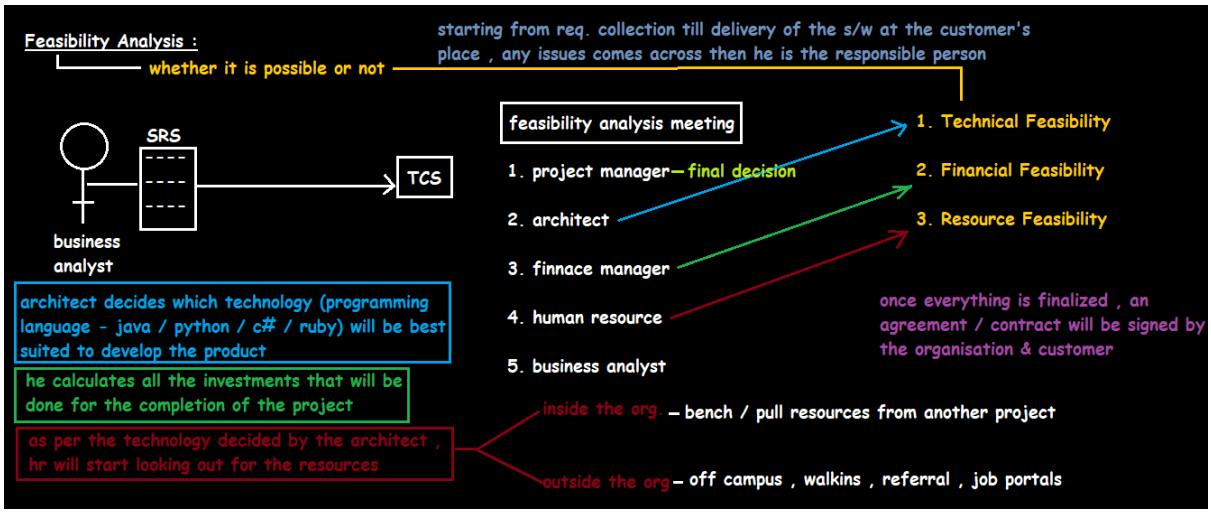
Business analyst is a person who is working in Service based company.







Feasibility Study:



Here we check whether, it is technically and financially possible to develop the software or not.

- done by software team consisting of **project managers**, **business analysts**, **architects**, **finance manager** and **HR's**.

Architect: –is the person who tells whether the product can be developed and if yes, then which technology is best suited to develop it.

- Here we check for,
    - Technical feasibility
    - Financial feasibility
    - Resource feasibility

Project Manager:

If any kind of complications arises in between the project then project manager is the responsible person.

Finance Manager:

He is the person responsible to calculate all the investments the organisation is going to make for the entire software development.

Human Resource Manager:

He / She looks for resources inside as well as outside the organisation.

Design:-

There are 2 stages in design,

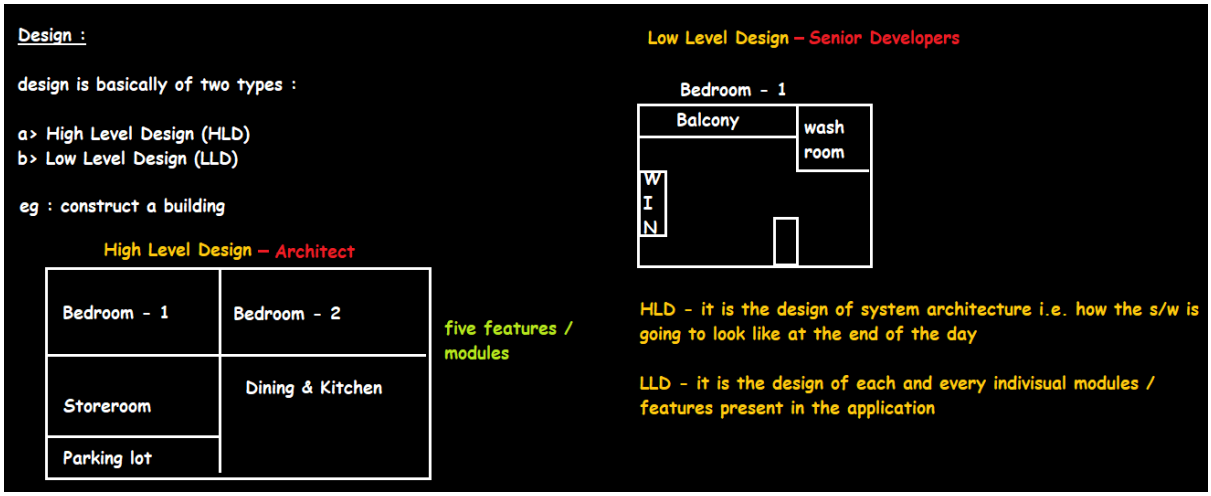
HLD – High Level Design

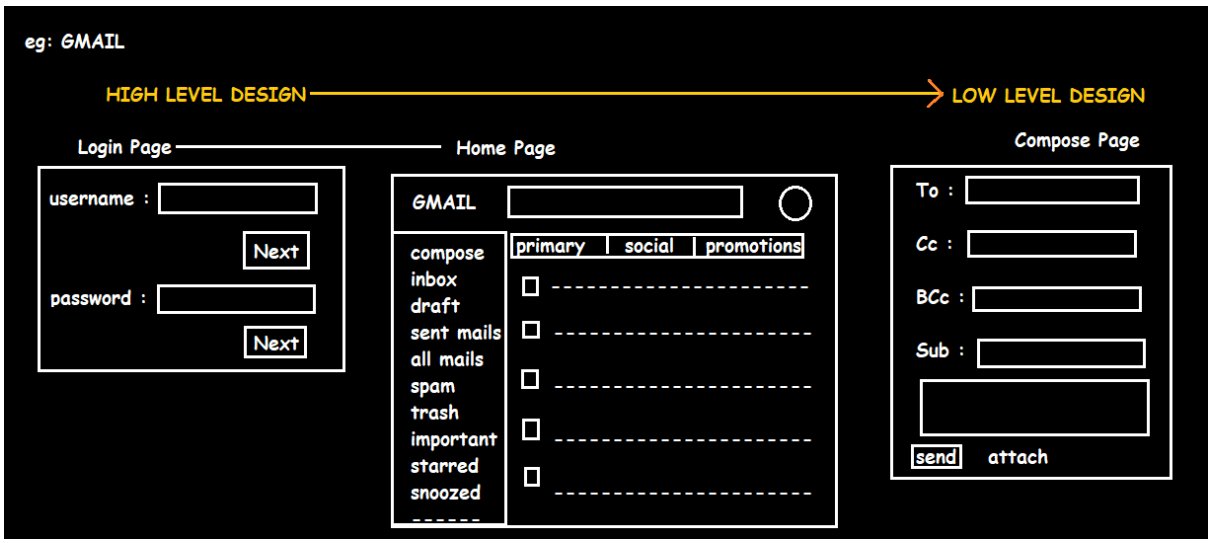
LLD – Low Level Design

HLD – gives the architecture of the software product to be developed and is done by **architects**

LLD – done by **senior developers**. It describes how each and every feature in the product should work and how every component should work. Here, only the design will be there and not the code.

For ex, let us consider the example of building a house.

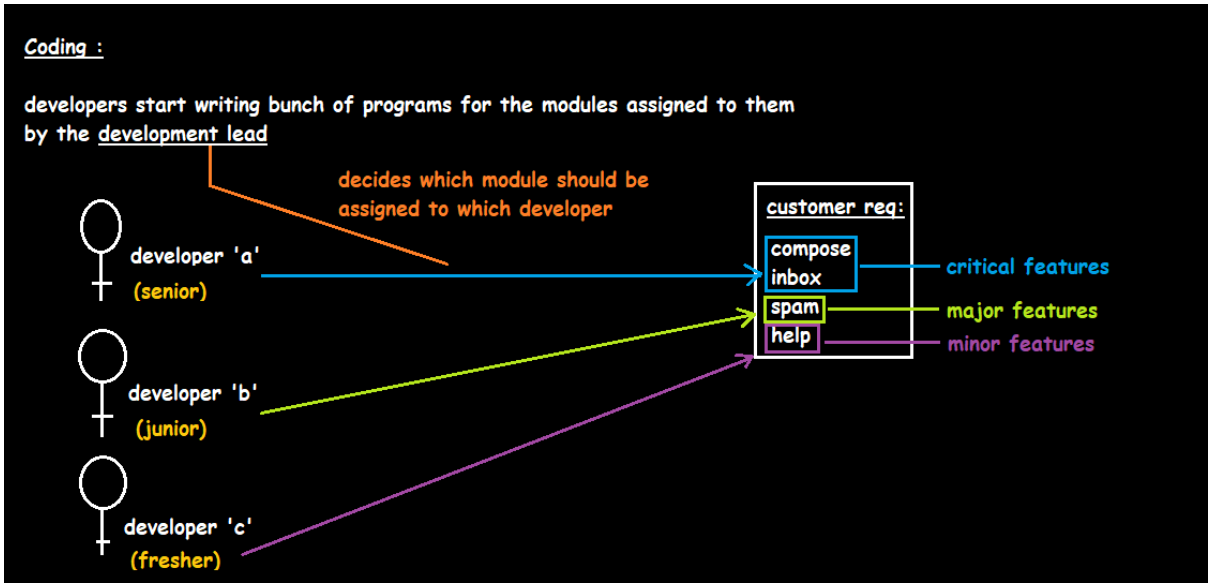




Coding / Programming:-

- Done by all developers – seniors, juniors, fresher's

- This is the process where we start building the software and start writing the codes for the product.



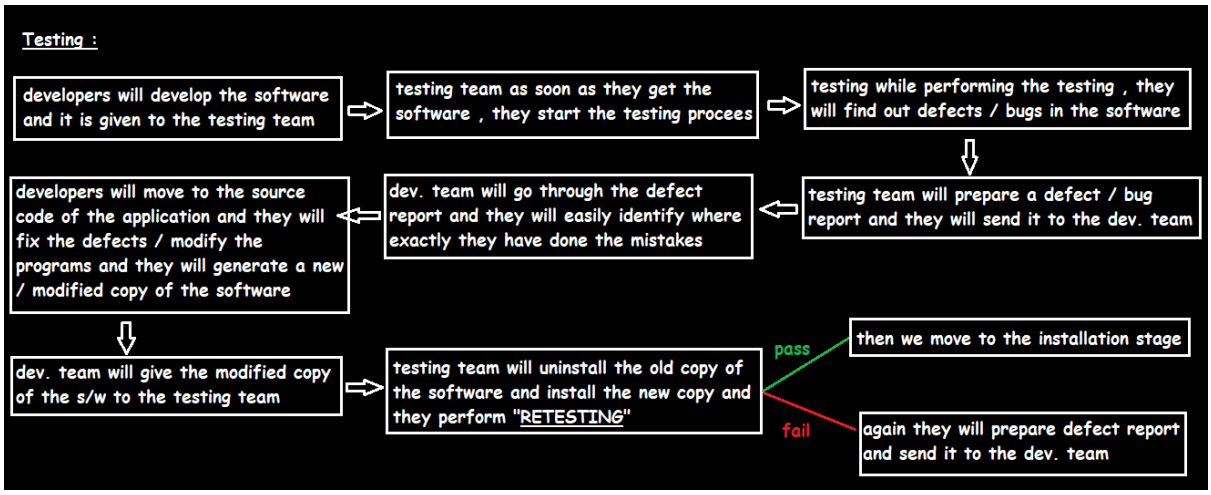
Testing:-

- done by test engineers

- It is the process of checking for all defects and communicating the same with the development team.

Once, test engineers find any defect, they will prepare defect report and send it to the developers. Developers then fix the defects by referring the defect report and send the new copy of the software to the testing team. Then the testing team will uninstall the old copy of the software and install the new copy of the software and they retest the software to check whether the earlier defects are fixed or not.

Once they come to know that the earlier defects are fixed, they inform the same to the installation engineers.



Installation:-

- done by installation engineers. They go to the clients place and install the software and give them a demo how to install the software in future.

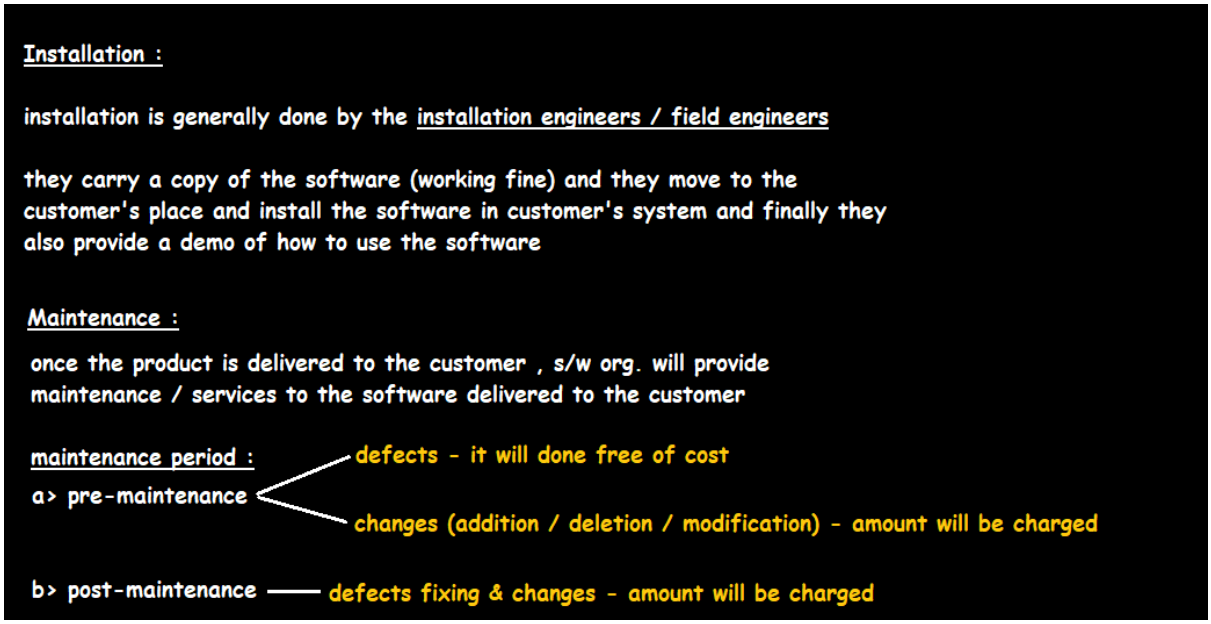
- For ex, consider the example of a software to be developed and installed at Reliance petrol bunk.

Maintenance:-

- Here as the customer uses the product, he finds certain bugs and defects and sends the product back for error correction and bug fixing.

- Bug fixing takes place

If within the maintenance period customer finds any defects then the software company will fix the defects free of cost. Post maintenance period even if customer find any defects, he /she will be charged for a single bug fix also.



100 % testing is not possible – because, the way testers test the product is different from the way customers use the product.

Service – based companies and Product – based companies

Service – based companies: -

They provide service and develop software for other companies

They provide software which is and specified as per the client company's requirement and never keep the code of the developed product and does not provide the software to any other company other than the client company.

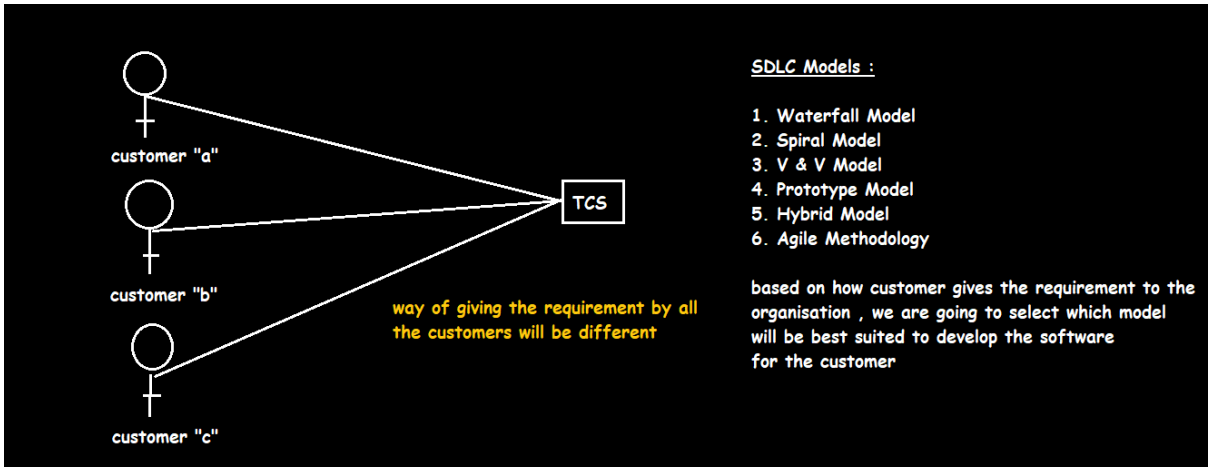
Ex – Wipro, Infosys, TCS, Accenture

Product – based companies:-

The develop software products and sell it to many companies which may need the software and make profits for themselves

They are the sole owners of the product they develop and the code used and sell it to other companies which may need the software.

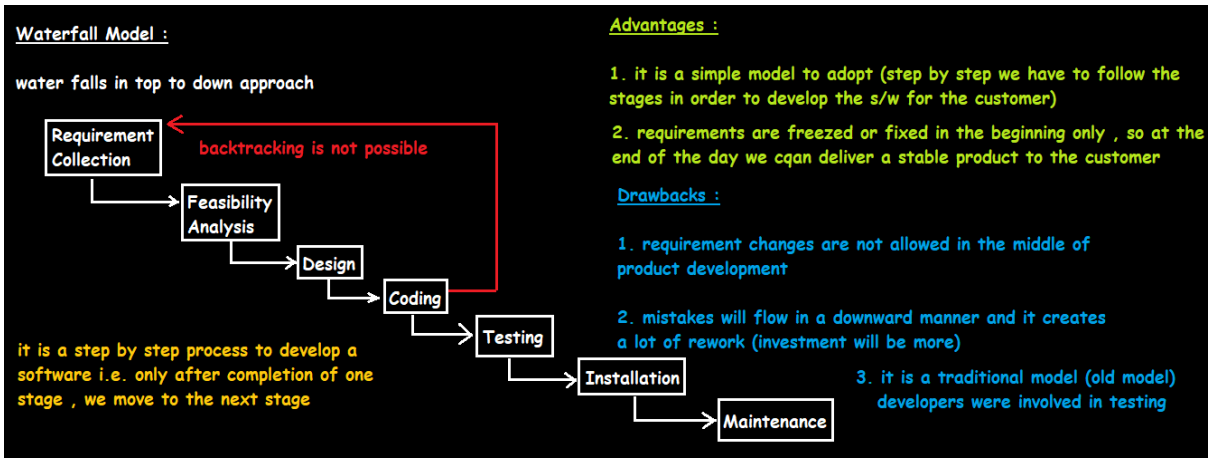
Ex – Oracle, Microsoft



Waterfall Model:

It is a step by step procedure to develop a software.

It is a traditional model.

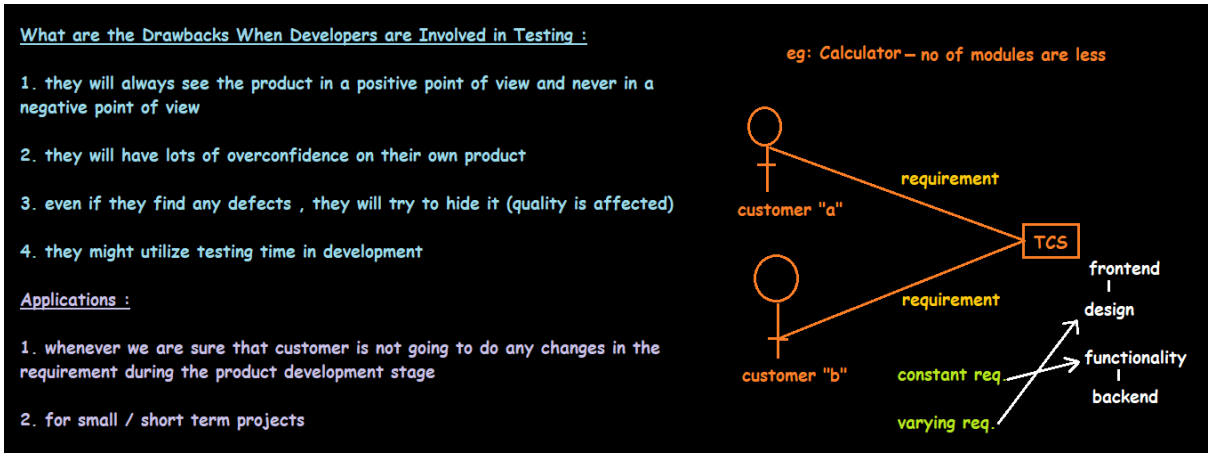


Advantages of waterfall model:

1. As in the beginning only requirements are fixed, so at the end of the day we will get a stable product.
2. It is a simple model to adopt.

Drawbacks of Waterfall Model:-

1. Backtracking is not possible. (We cannot go back and change requirements once the design stage is reached. Thus the requirements are freezed once the design of the software product is started). So at the end of the day we can deliver a stable or quality product.
2. Developers are involved in testing.
3. Requirement is not tested, design is not tested, if there is a bug in the requirement, it goes on till the end and leads to lot of rework and the investment done on the project will be more.



Applications of waterfall model:-

Used in – developing a simple application

- For short term projects

- Whenever we are sure that the requirements will not change

For **ex**, waterfall model can be used in developing a simple calculator as the functions of addition, subtraction etc. and the numbers will not change for a long time.

Interview Questions on Waterfall Model :

1. Explain Waterfall Model
2. What are the advantages , drawbacks & applications if we adopt Waterfall model
3. What are the drawbacks if developers are involved in the testing process
4. What happens in the feasibility analysis stage
5. What are the roles & responsibilities of Project Manager , Finance Manager & Architect
6. What is the difference between HLD & LLD
7. How the installation stage / process works
8. What happens in the Maintenance stage
9. What is the role of Business Analyst in a service based company & role of a Product Analyst in a product based company
10. What is the difference between SBC & PBC

Spiral Model:-

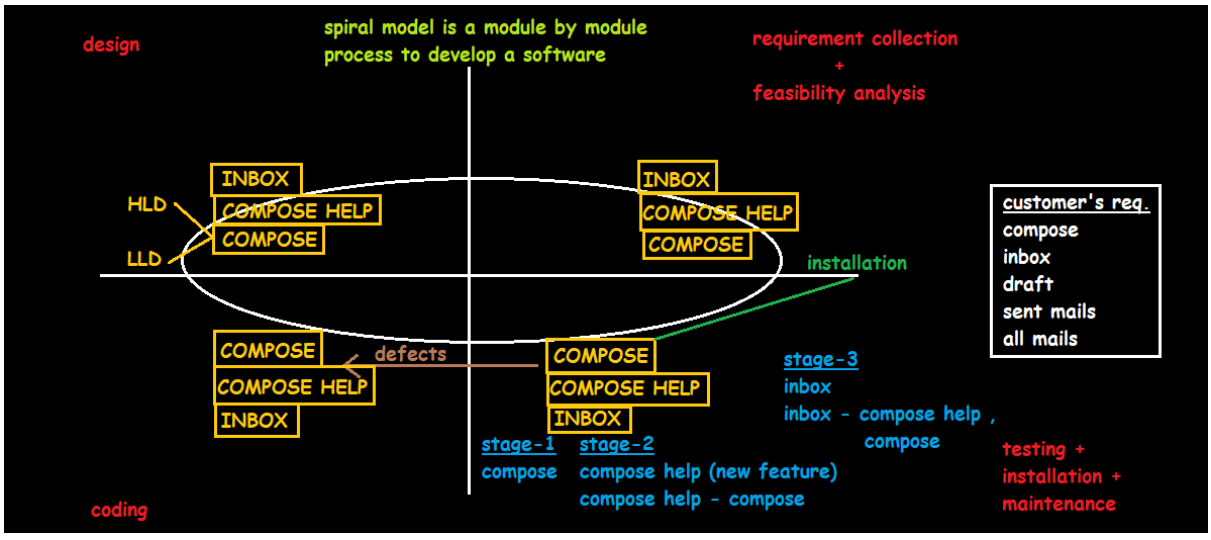
Spiral Model :

in waterfall model , customer used to give the complete requirement to the software organisation for the development of the software.

in spiral model , customer will give the requirements in stages or in module wise. only after completion of one module , customer will give the requirement of next module.

in spiral model now we can handle requirement changes in the middle of product development. customer need not wait for the complete software to be developed in order to do any kind of changes.

It is a module by module process to develop a software.



Advantages of Spiral Model:-

1) Requirement changes are allowed in the middle of product development.

We do requirement changes in 2 ways:

1. Major change (Addition, Deletion and Modification of any features)
2. Minor change (Bug Fixes)

2) After we develop one feature / module of the product, then and only then we can go on to develop the next module of the product.

3) Customer gets an idea how his product will look like at the end of the day.

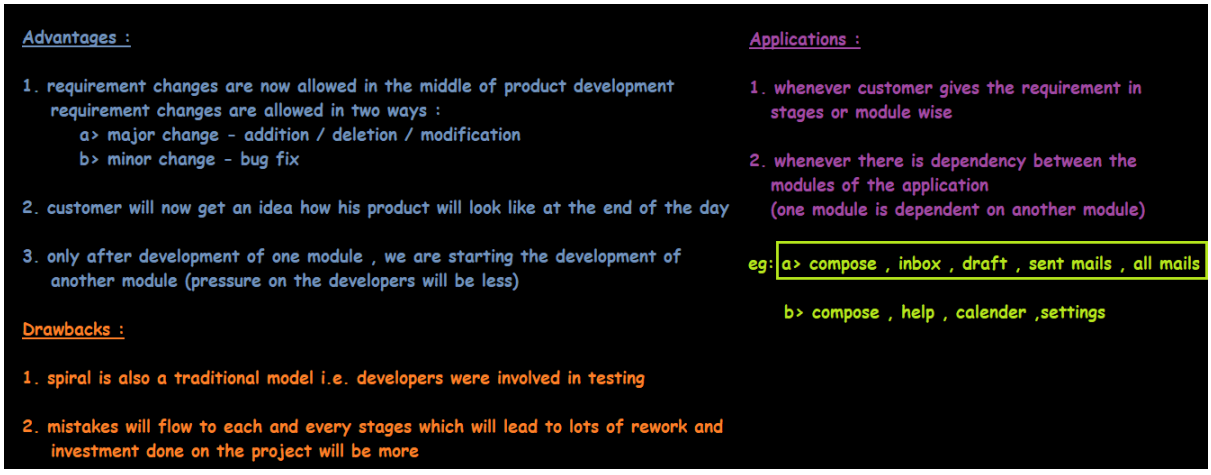
Drawbacks of Spiral Model:-

1. Downward flow of defects will be there which creates lots of rework and investment done in the project will be more.
2. Developers are involved in testing.

Applications of Spiral Model:-

1. Whenever there is dependency in building the different modules of the software, then we use Spiral Model.

2. Whenever the customer gives the requirements in stages, we develop the product in stages.



Interview Questions on Spiral Model :

1. explain how spiral model works
2. explain the advantages , drawbacks & applications of spiral model
3. why spiral model is called as a module by module process to develop a s/w
4. how spiral model is different from waterfall model

3) V – MODEL / V & V MODEL (Verification and Validation Model)

V & V Model : (Verification & Validation Model)

this model was introduced inorder to overcome the drawbacks of Waterfall and Spiral model

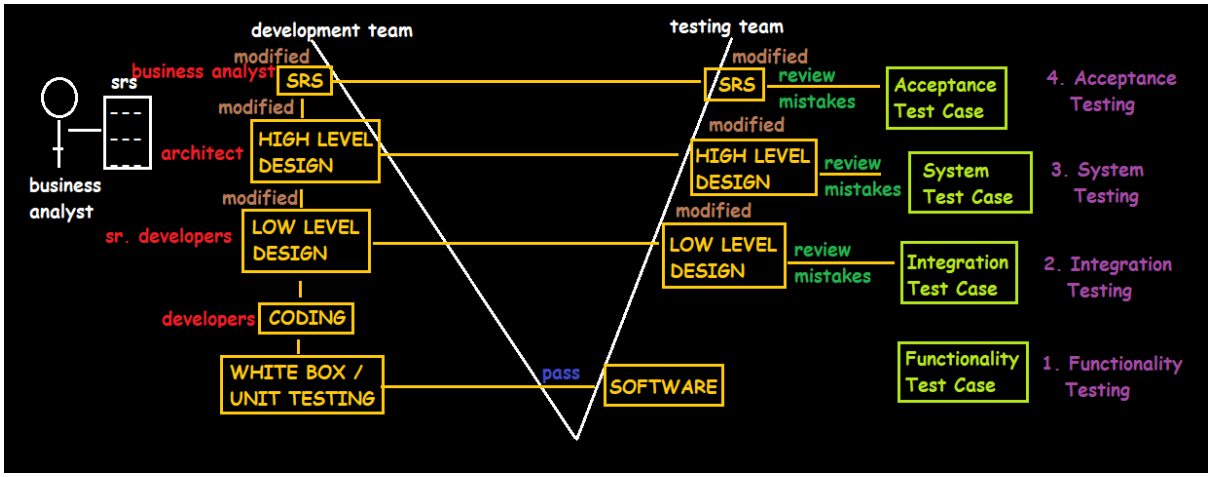
drawbacks of waterfall & spiral model :

1. traditional model (developers were involved in testing)
2. downward flow of mistakes ---> rework was there ---> investment was high

overcome :

1. we have a separate testing team involved for the testing process
2. we start the testing from the initial stage only and all the stages are tested

This model came up in order to overcome the drawback of waterfall model – here testing starts from the requirement stage itself where in Waterfall and Spiral model testing was done only after the coding stage.



Advantages of V&V model

1) Testing starts in very early stages of product development which avoids downward flow of defects which in turn reduces lot of rework

2) Testing is involved in every stage of product development

3) Total investment is less – as there is no downward flow of defects. Thus there is less or no re-work

Drawbacks of V&V model

1) Initial investment is more – because right from the beginning testing team is needed

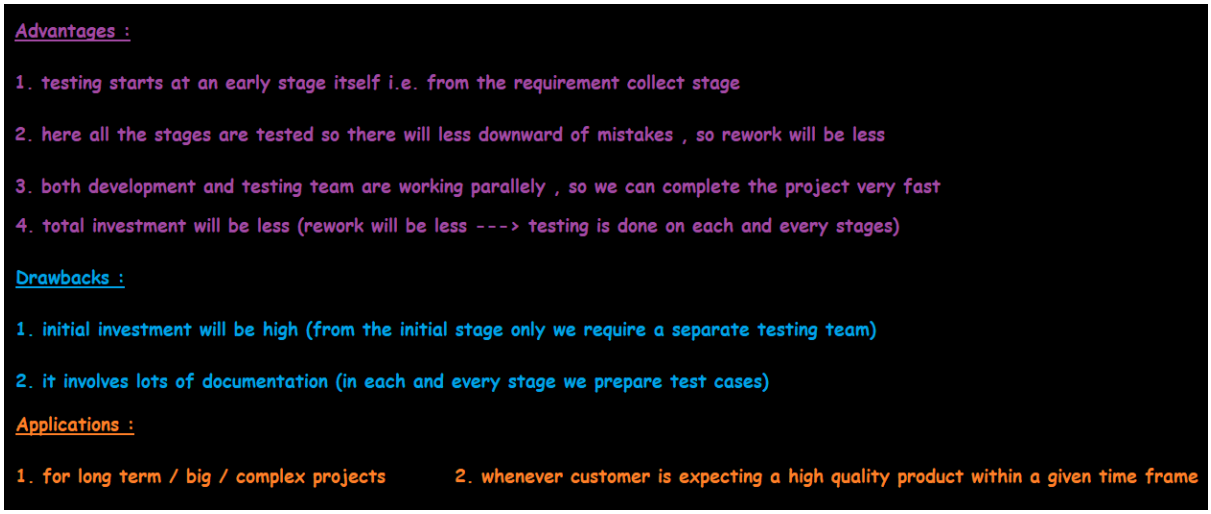
2) More documentation work – because of the test cases and all other documents that is prepared.

Applications of V&V model

We go for V&V model in the following cases,

1) For long term / big / complex projects

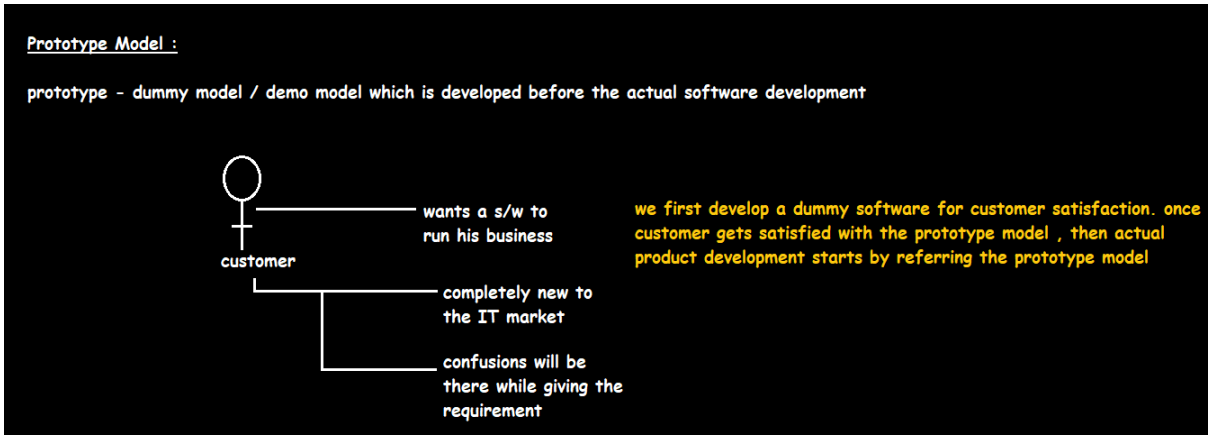
2) When customer is expecting a very high quality product within stipulated time frame

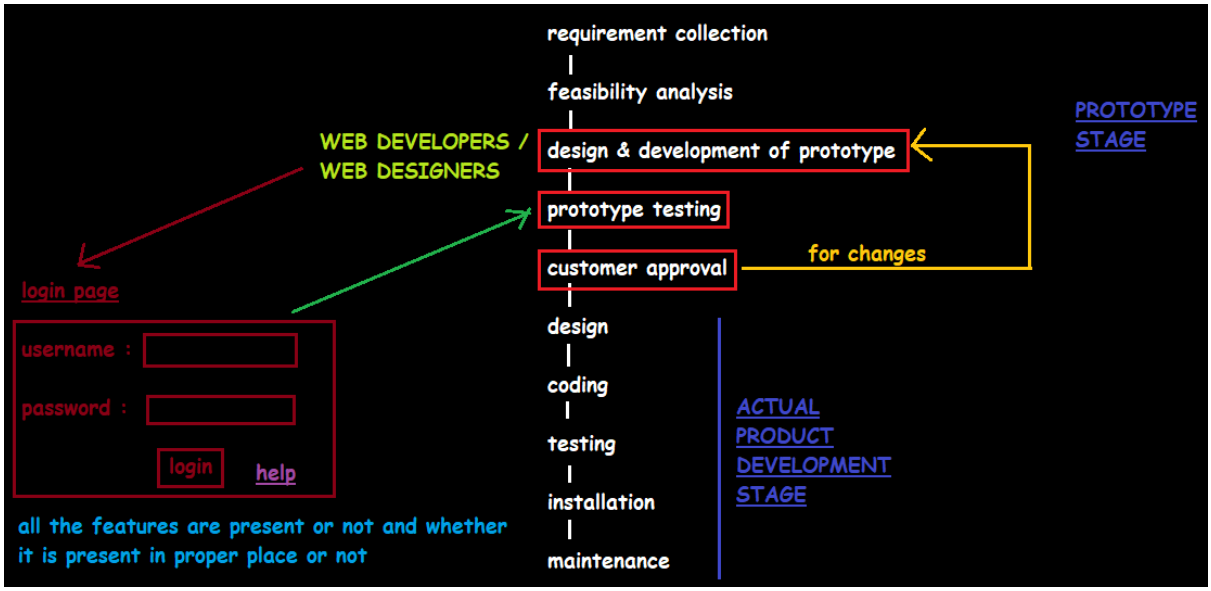


Interview Questions on V & V Model :

1. What is V & V model - it is a step by step process to develop a s/w
2. Explain how V & V model works
3. How V & V model is different from Waterfall & Spiral model
4. What are the advantages , drawbacks and applications of V & V model

4) PROTOTYPE DEVELOPMENT MODEL





The requirements are collected from the client in a textual format. The prototype of the s/w product is developed. The prototype is just an image / picture of the required s/w product. The customer can look at the prototype and if he is not satisfied, then he can request more changes in the requirements.

Prototype testing means testers are checking if all the components mentioned are existing.

The difference b/w prototype testing and actual testing – in PTT, we are checking if all the components are existing or not and whether all the components are in right place or not, whereas, in ATT we check if all components are working functionally or not.

Advantages of Prototype model

1) In the beginning itself, we set the expectation of the client.

2) There is clear communication b/w development team and client as to the requirements and the final outcome of the project is good.

3) Major advantage is – customer gets the opportunity in the beginning itself to ask for changes in requirements as it is easy to do requirement changes in prototype rather than real applications. Thus expectations are met easily in the beginning only.

4) Customer can ask for changes in the beginning itself.

Drawbacks of Prototype model

1) There is delay in starting the real project development

2) To improve the communication, there is an investment needed in building the prototype.

Applications of Prototype Model

We use this model when,

1) When customer is new to the s/w industry

2) When customer is not clear about his / her own requirements

Advantages :

1. customer gets an idea how his product is going to look like at the end of the day

2. customer can ask for "n" no of changes before the actual product development

3. there is a frequent communication between the customer and the development team

4. development team can set the expectation of the customer in the beginning only

Drawbacks :

1. there is a lot of delay in starting the actual product development

2. there is an investment needed for the communication between the customer and the dev. team

Applications :

1. whenever customer is new to the s/w industry 2. whenever customer is not sure about his own requirements

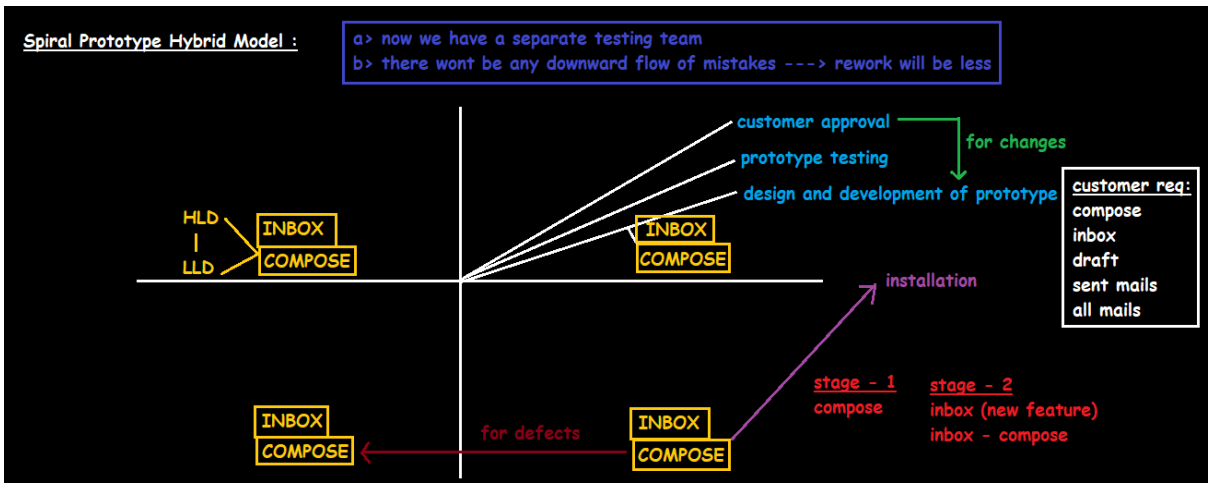
Interview Questions on Prototype Model:

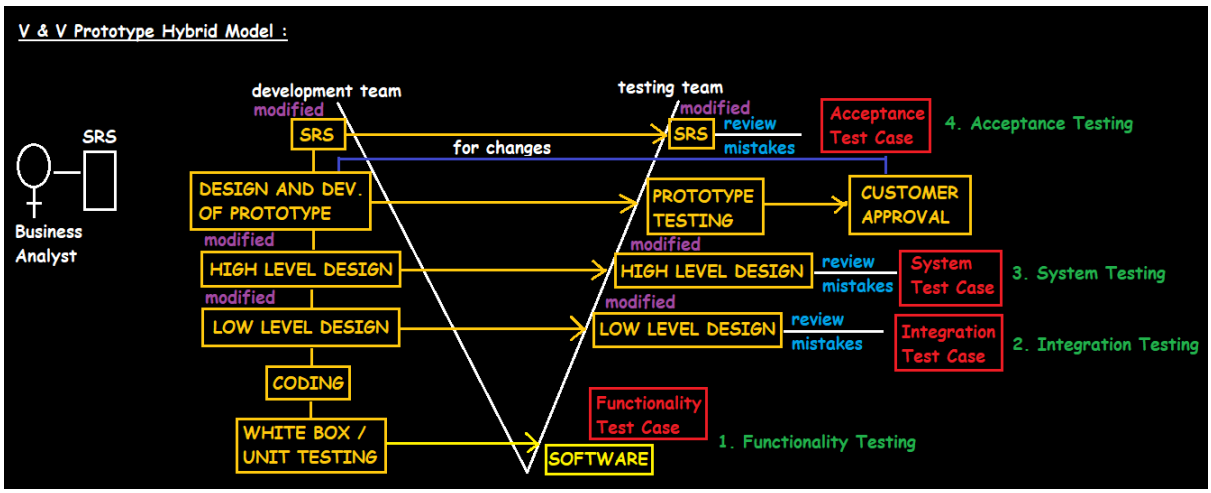
1. Explain prototype model / explain how prototype model works
2. difference between prototype & actual testing
3. advantages , drawbacks & application of prototype model
4. why we generally go for prototype model

5) HYBRID MODEL

It is the combination of two models and it is of two types:

1. Spiral Prototype Hybrid Model
2. V & V Prototype Hybrid Model



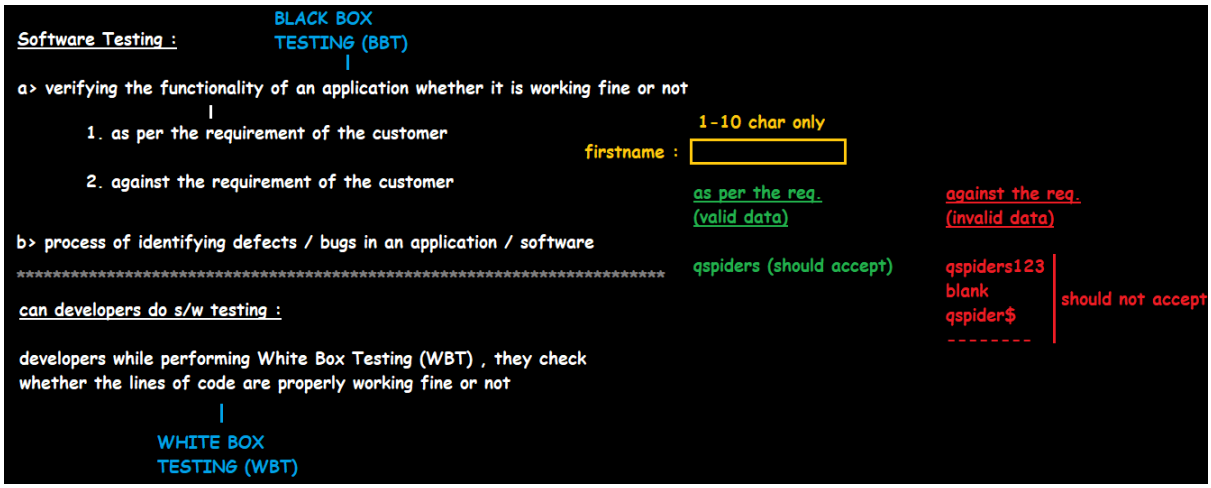


Definition of Software Testing

It is a process of finding or identifying defects in s/w is called s/w testing.

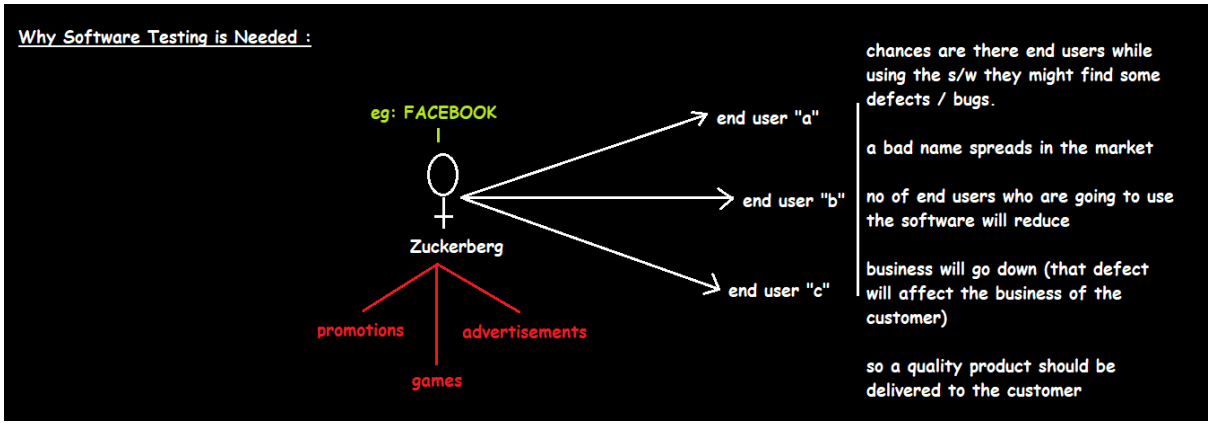
It is verifying the functionality (behaviour) of the application(s/w) as per and against the requirements specification.

It is the execution of the s/w with the intention of finding defects. It is checking whether the s/w works according to the requirements.



Why Software Testing is needed?

Every software is developed to support customers business. If we are not testing the software and directly deploying it at the clients place, chances are there client may find a lot of defects which in turn may affect his business. SO bad name spreads in the market and no of users who are going to use the software will be less. So before delivering the software to the client, testers test the software, find the defects and get it fixed by the development team and then give it to the client.

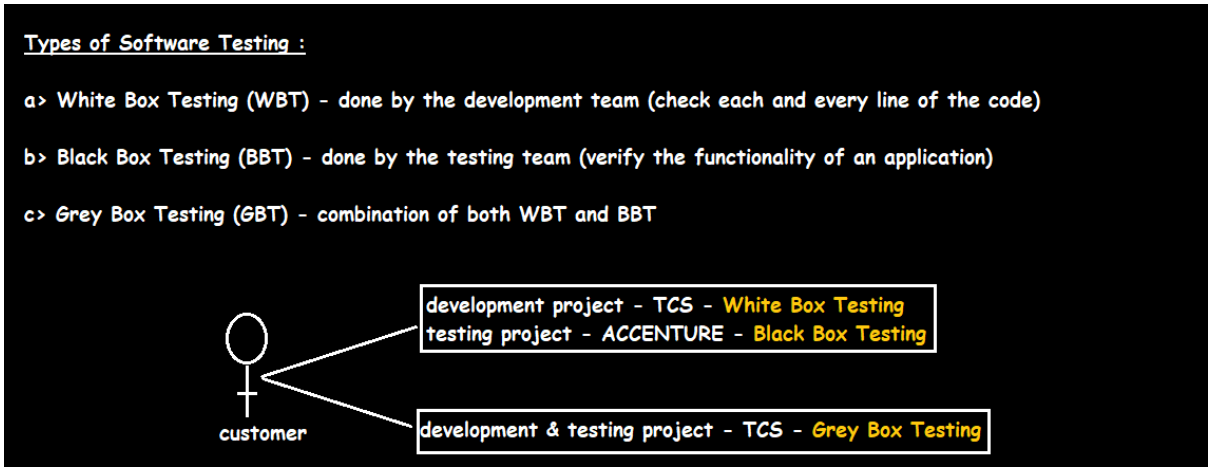


There are 3 types of s/w testing, namely,

1) **White box testing** – also called unit testing or glass box testing or transparent testing or open-box testing

2) **Black box testing** – also called as functional testing or open box testing

3) **Grey Box Testing** – It is a combination of both WBT and BBT



Difference between White Box Testing and Black Box testing

1) White Box Testing 2) Black Box Testing

a) 1) Done by developers

2) Done by test engineers

b) 1) Look into the source code and test the logic of the code

2) Verifying the functionality of the application as per and against requirement specifications

c) 1) should have knowledge of internal design of the code

2) No need to have knowledge of internal design of the code

d) 1) should have knowledge of programming

2) No need to have knowledge of programming

e) 1) WBT is also called as Unit/Glass box/ Open box/Transparent testing

2) BBT is also called as closed box/ Functional testing.

WHITE BOX TESTING (WBT)	BLACK BOX TESTING (BBT)
a> done by the development team	a> done by the testing team
b> they check each and every lines of the code whether it is working fine or not	b> they check the functionality of the application as per and against the requirement of the customer
c> to perform white box testing , programming knowledge is required	c> to perform black box testing , no programming knowledge is required
d> to perform white box testing , internal code design (hld & lld) knowledge is required	d> to perform black box testing , no internal code design knowledge is required
e> white box testing is also called as unit (each and every unit of the code is tested) / openbox / glassbox / transparent (source code is visible to the developers)	e> black box testing is also called as closed box (source code is not visible to the testing team) / functional (we verify the functionality of the appln)

Types of White Box Testing :

a> Loop Testing
b> Conditional Testing
c> Path Testing

Types of Black Box Testing :

a> Functional Testing
b> Non-Functional Testing
   |
   1. Performance Testing
   2. Usability Testing
   3. Reliability Testing
   4. Recovery Testing
   5. Accessibility Testing

1. Functionality Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing
5. Smoke Testing

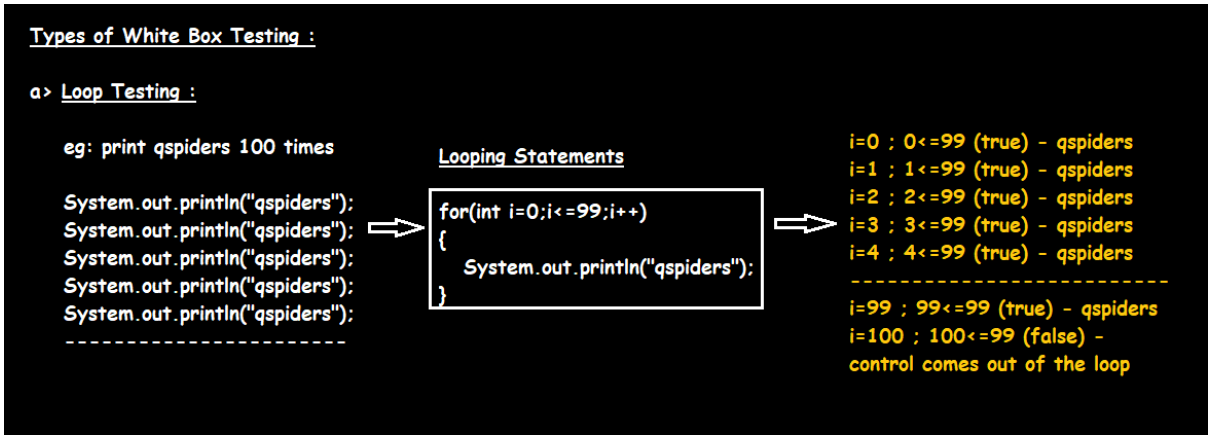
6. Adhoc Testing
7. Exploratory Testing
8. Regression Testing
9. Compatibility Testing
10 . Globalization Testing

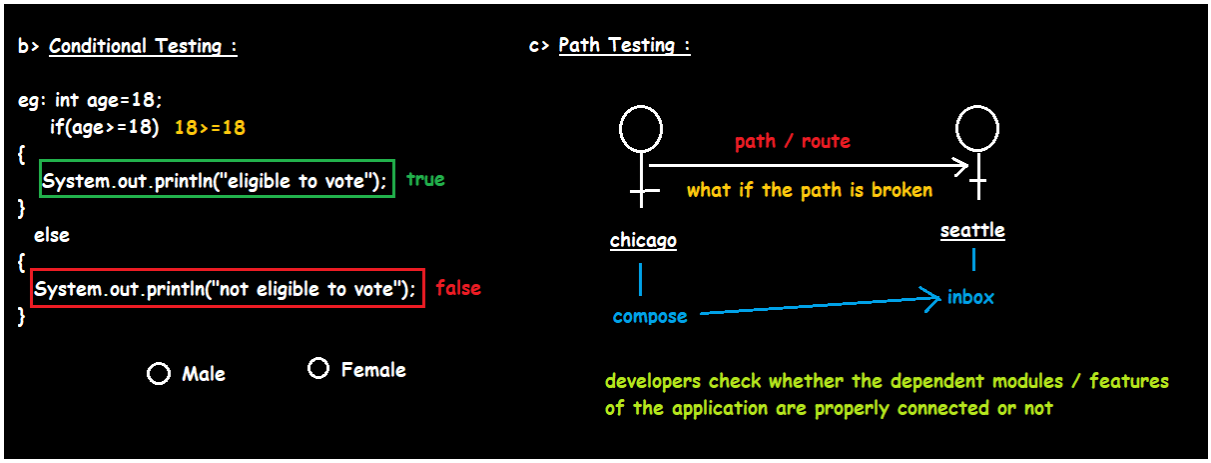
WHITE BOX TESTING (WBT)

Entire WBT is done by developers. It is the testing of each and every line of code in the program. Developers do WBT, sends the s/w to testing team. The testing team does black box testing and checks the s/w against requirements and finds any defects and sends it to the developer. The developers fixes the defect and does WBT and sends it to the testing team. Fixing defect means the defect is removed and the feature is working fine.

Types of White Box Testing

1. Path Testing
2. Loop Testing
3. Conditional Testing





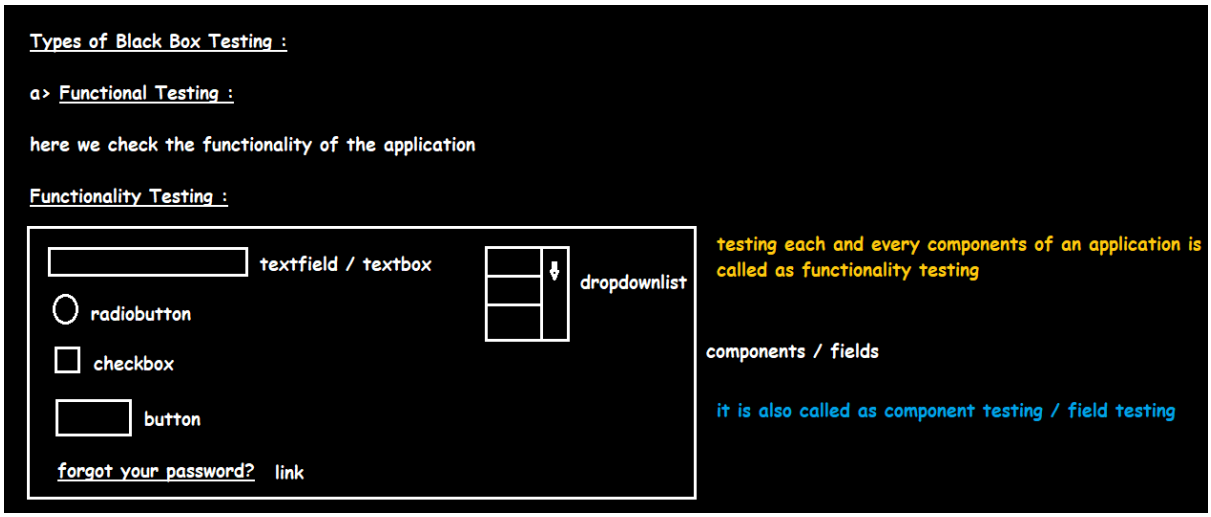
BLACK BOX TESTING (BBT)

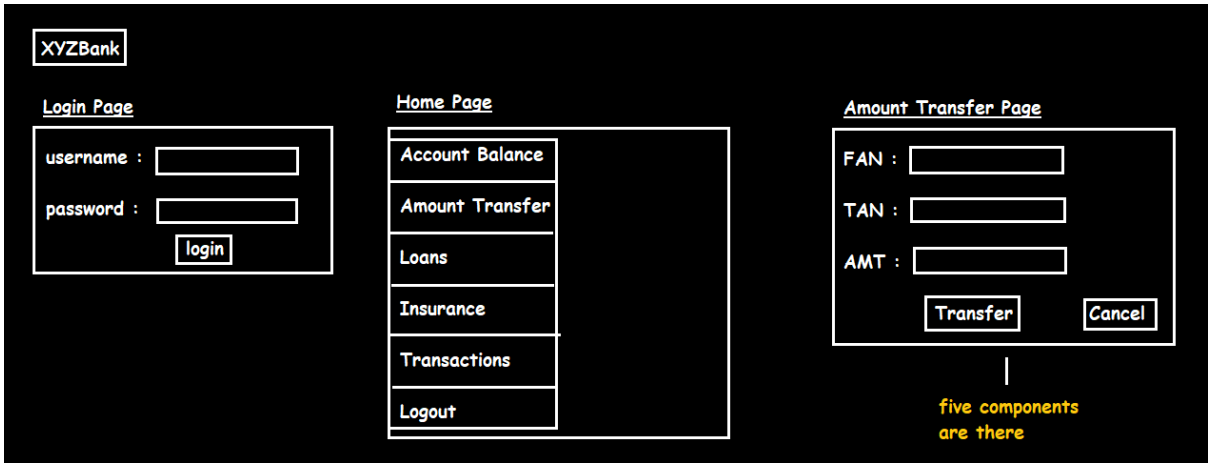
It is verifying the functionality (behaviour) of an application according as well as against requirement specifications.

Types of Black Box Testing

1) FUNCTIONALITY/COMPONENT/FIELD TESTING

Testing each and every component thoroughly (rigorously) is known as component testing.



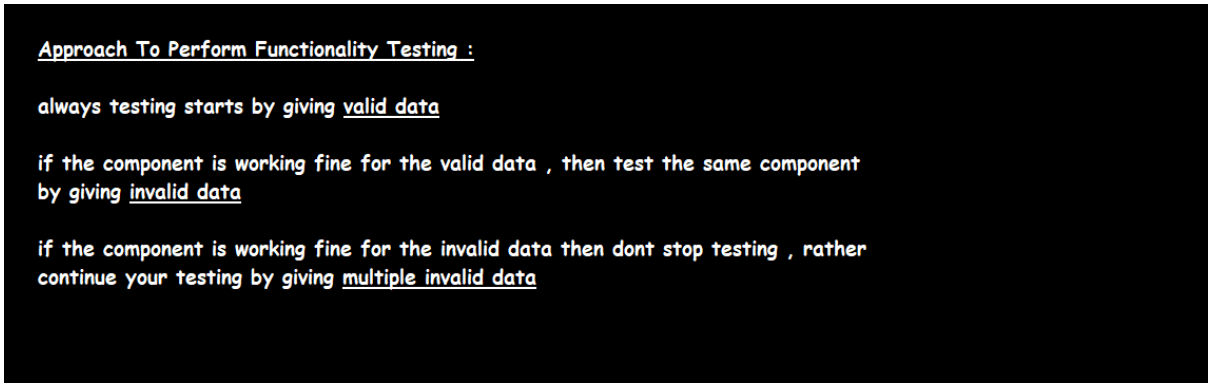


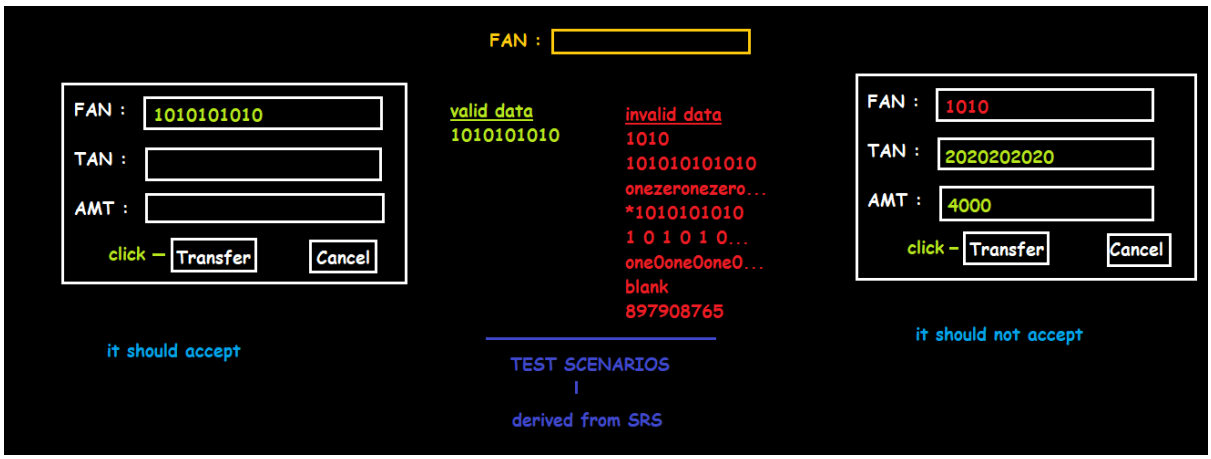
Procedure to do Component Testing

Thus, during testing, we must remember the following points,

Always testing starts by giving valid data. If the component is working fine for the valid data, we should stop testing and start testing the same component by giving invalid data.

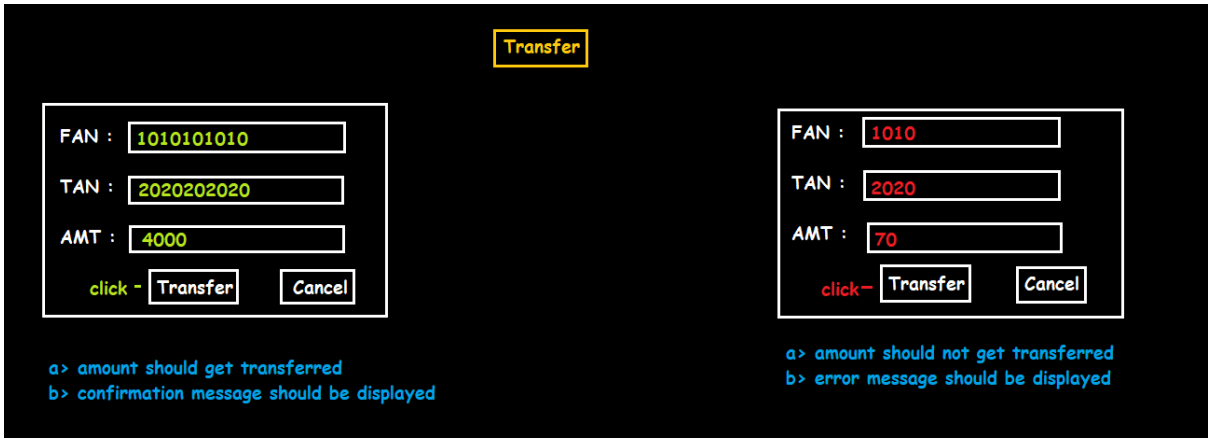
If the component is working for one invalid data we should not stop testing, we should continue testing the same component by giving some more invalid data's.

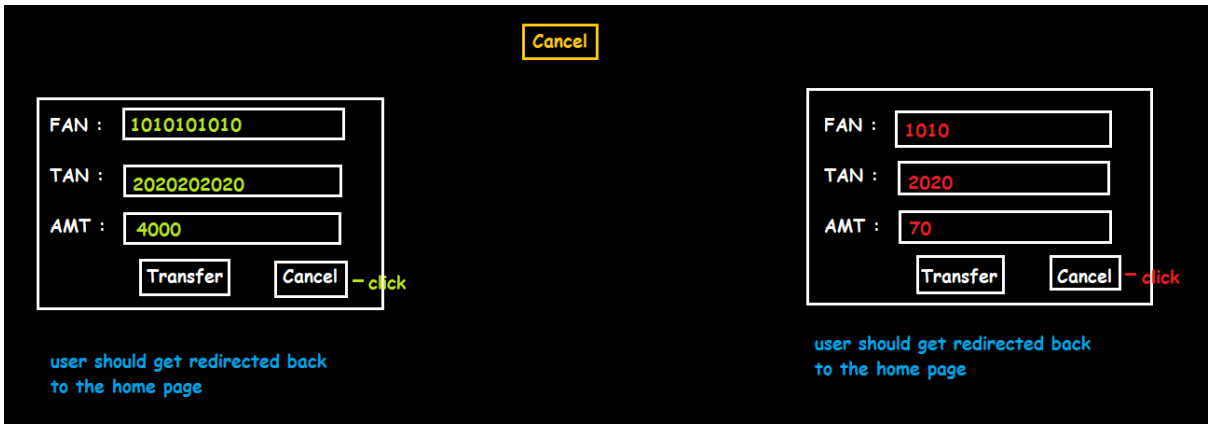


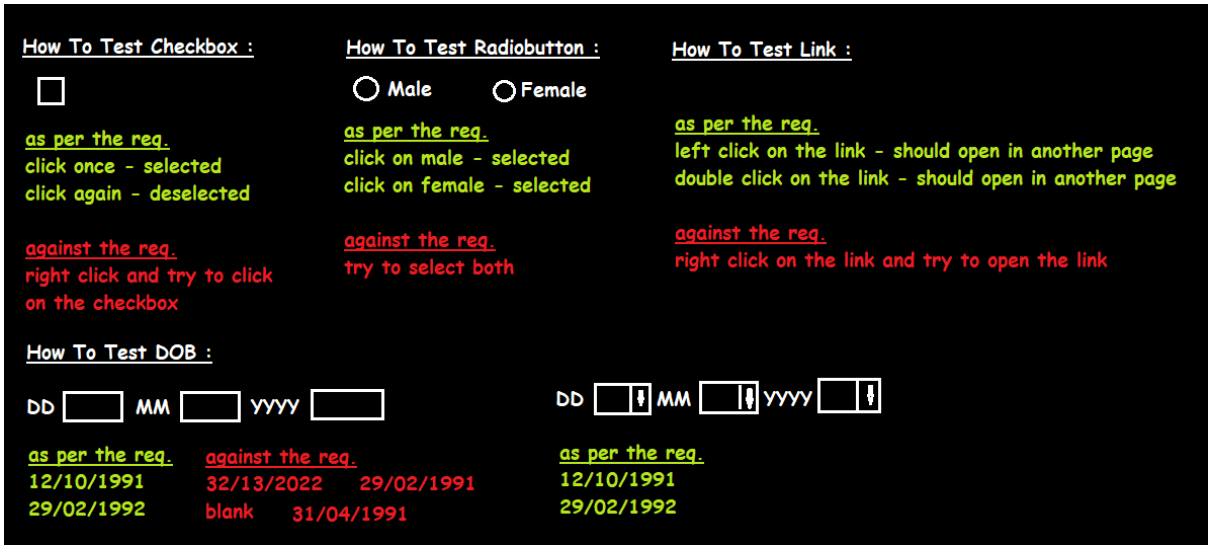


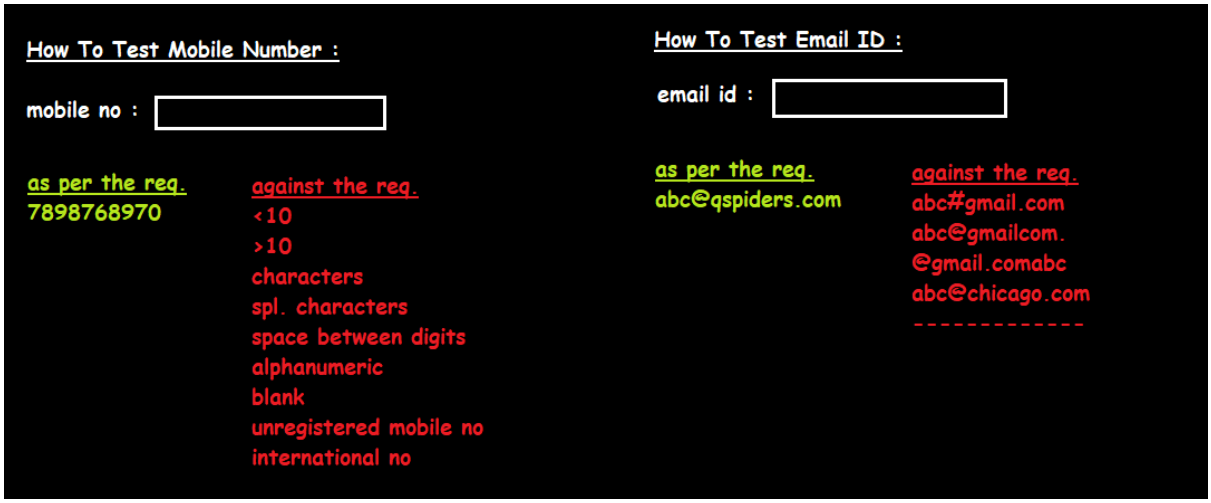
valid data	invalid data
2020202020	2020
	202020202020
	twozerotwozero...
	*2020202020
	blank
	2 0 2 0 2 0...
	two0two0two0...
	6789056789
	1010101010 (FAN)

valid data	invalid data
4000	70
	7000
	$4000
	fourthousand
	4 0 0 0
	four000
	4000.50
	blank









We must not do **over-testing** (testing for all possible junk values) or **under-testing** (testing for only 1 or 2 values). We must only try and do **optimize testing** (testing for only the necessary values- both invalid and valid data).

We must do both positive testing (testing for valid data) and negative testing (testing for invalid data).

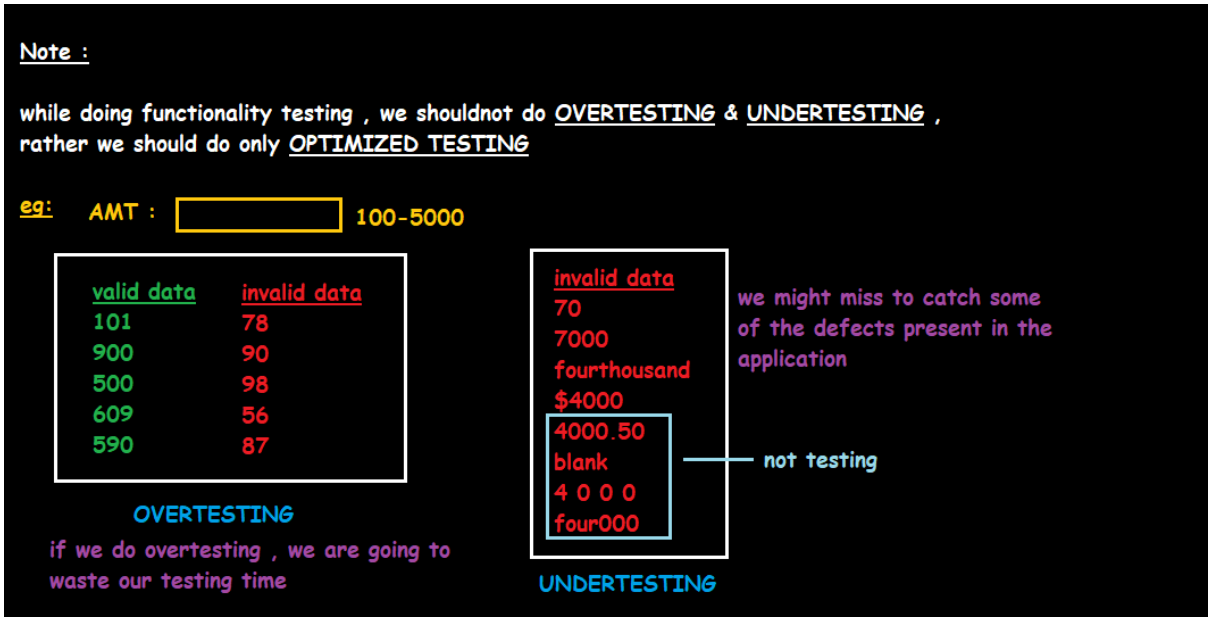
Over Testing- Testing the components of an application by giving data which does not make any sense i.e. junk data.

If we are ding over testing, then we are wasting our testing time.

Under Testing- Testing the components of an application by giving insufficient set of scenarios.

If we are doing under testing, then we might miss to catch some defects in the application.

Optimized Testing- Testing the components of an application by giving data which makes sense.



Types of Functionality Testing

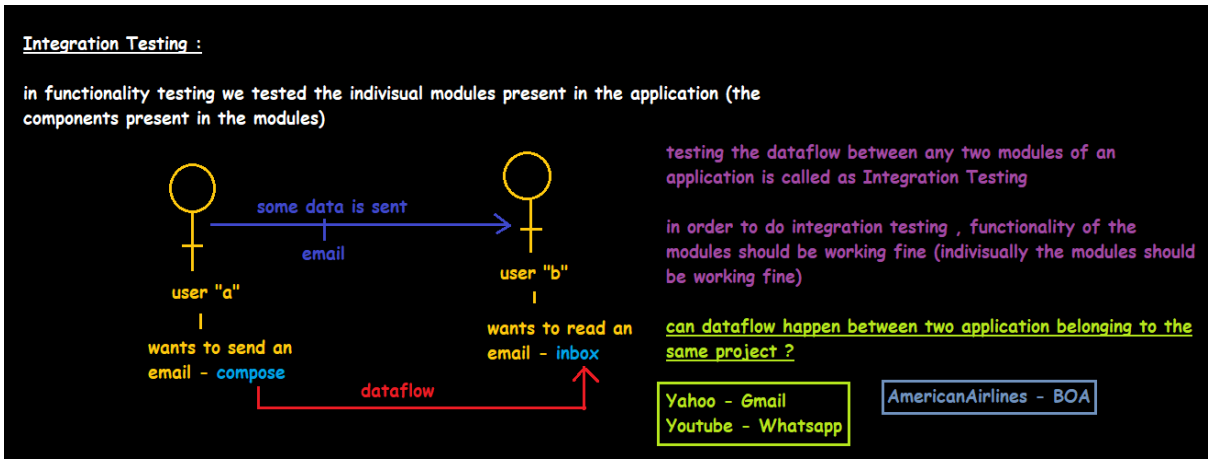
1. **Positive Functionality Testing**
2. **Negative Functionality Testing**

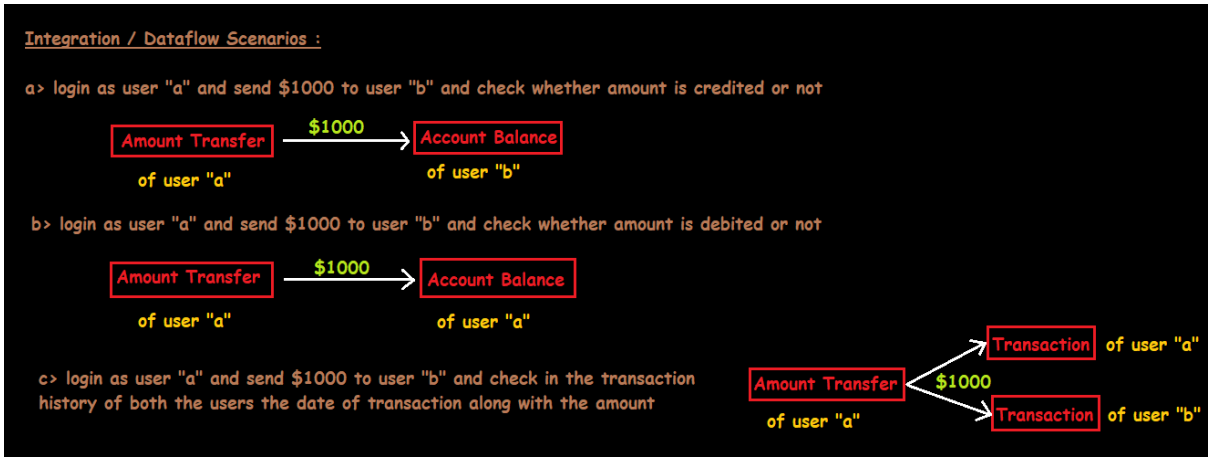
Testing the components of an application by giving positive or valid data is called as Positive Functionality Testing.

Testing the components of an application by giving negative or invalid data is called as Negative Functionality Testing.

2) INTEGRATION TESTING

Testing the data flow between two features/ modules is known as integration testing.





Eg: Take 2 features A & B. Send some data from A to B. Check if A is sending data and also check if B is receiving data

Now let us consider the example of banking s/w

Scenario 1 – Login as A to amount transfer – send 1000rs amount – message should be displayed saying 'amount transfer successful' – now logout as A and login as B – go to amount balance and check balance – balance is increased by 1000rs – thus integration test is successful.

Scenario 2 – also we check if amount balance has decreased by 1000rs in A

Scenario 3 – click on transactions – in A and B, message should be displayed regarding the data and time of amount transfer

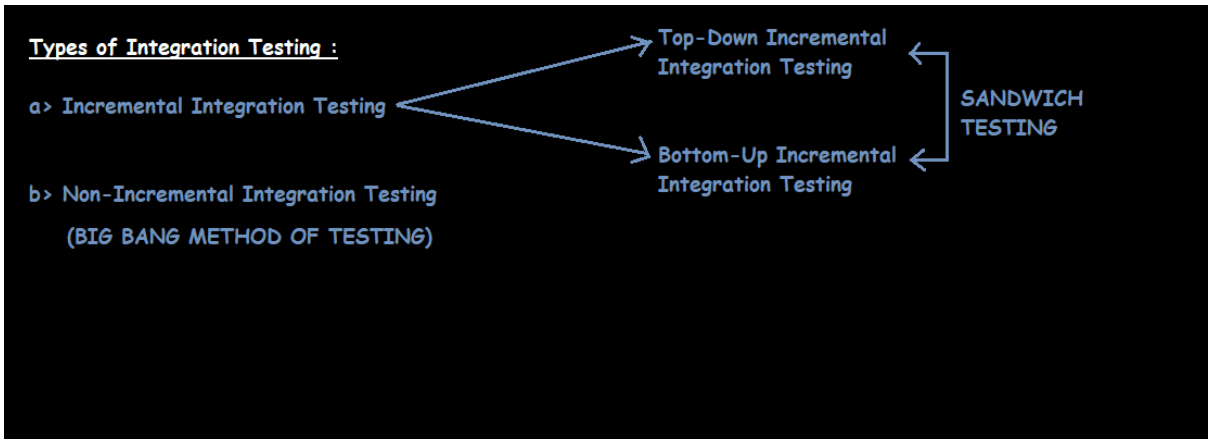
Let us consider Gmail software

We first do functionality testing for username and password and submit and cancel button. Then we do integration testing. The following scenarios can be considered,

Scenario 1 – Login as A and click on compose mail. We then do functional testing for the individual fields. Now we click on send and also check for save drafts. After we send mail to B, we should check in the sent items folder of A to see if the sent mail is there. Now we logout as A and login as B. Go to inbox and check if the mail has arrived.

There are two types of integration testing,

1. Integration testing
    1.1. Incremental Integration Testing
        1.1.1. Top-down Integration Testing
        1.1.2. Bottom-up Integration Testing
    1.2. Non-Incremental Integration Testing/ Big Bang Method of Testing



Incremental Integration Testing:

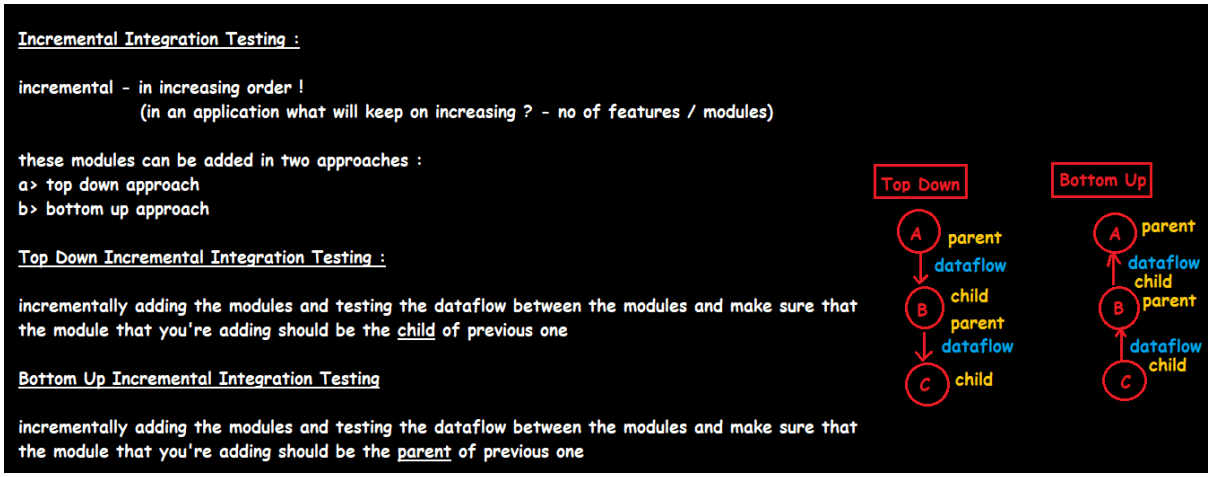
Incrementally add the modules the test the data flow between the modules is called incremental testing.

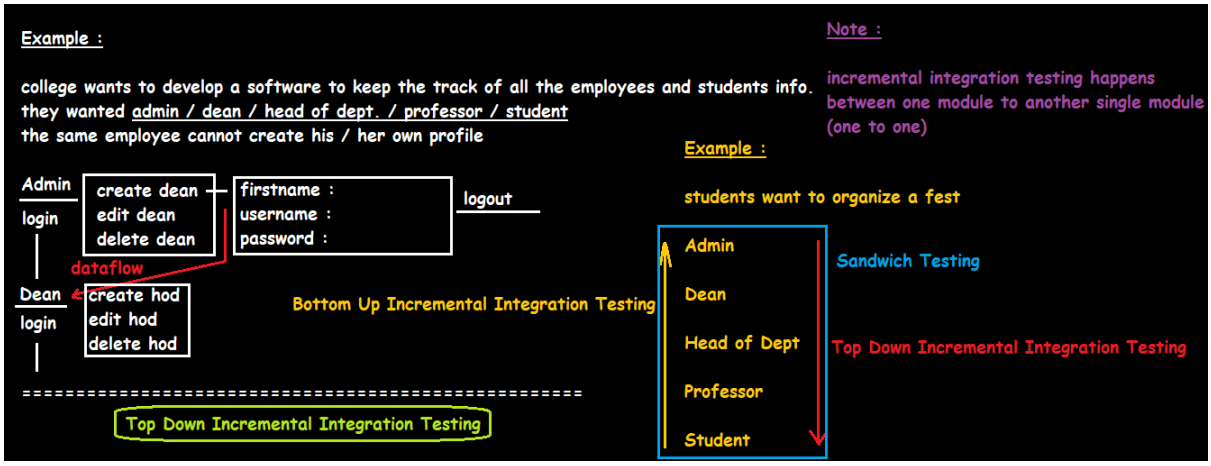
Top-down Integration Testing:

Incrementally add the modules the test the data flow between the modules and make sure that the module that you are adding is the child of previous one.

Bottom-up Integration Testing

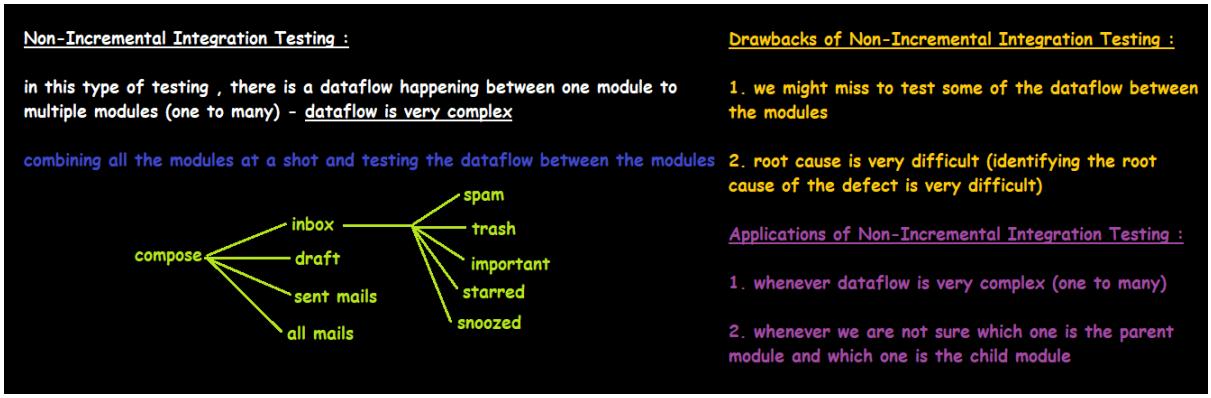
Incrementally add the modules the test the data flow between the modules and make sure that the module that you are adding is the parent of previous one.





Non-Incremental Integration Testing:

Combining all the modules at a shot and testing the data flow between the modules is called non- incremental integration testing.



Drawbacks of Non-Incremental Integration Testing:

1. We might miss to test some of the data flow between the modules.
2. Root cause analysis is very difficult i.e. identifying root of the defect is very difficult.

Applications of Non-Incremental Integration Testing:

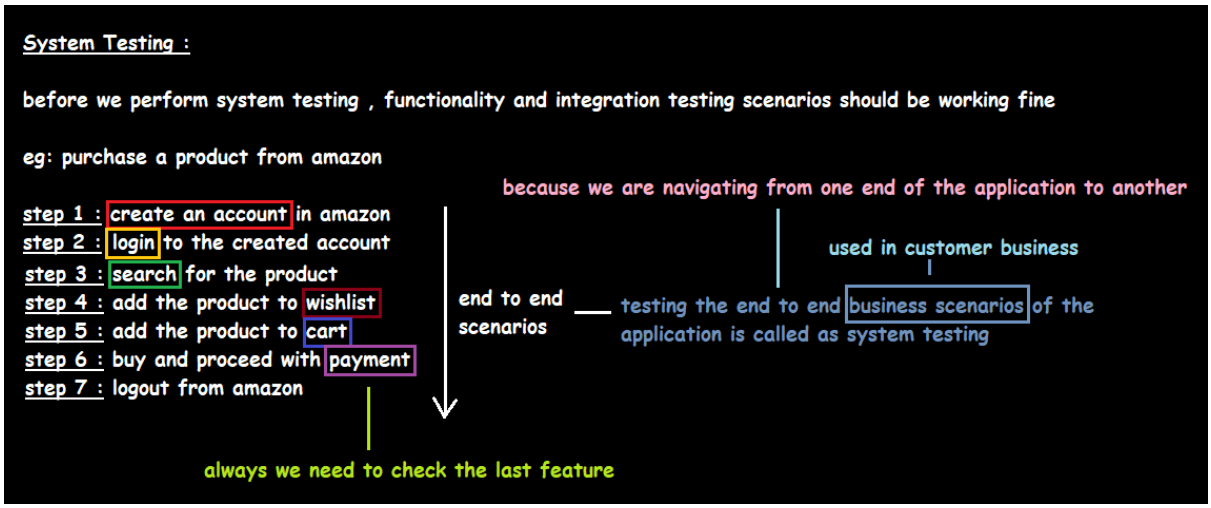
1. Whenever the dataflow is very complex.

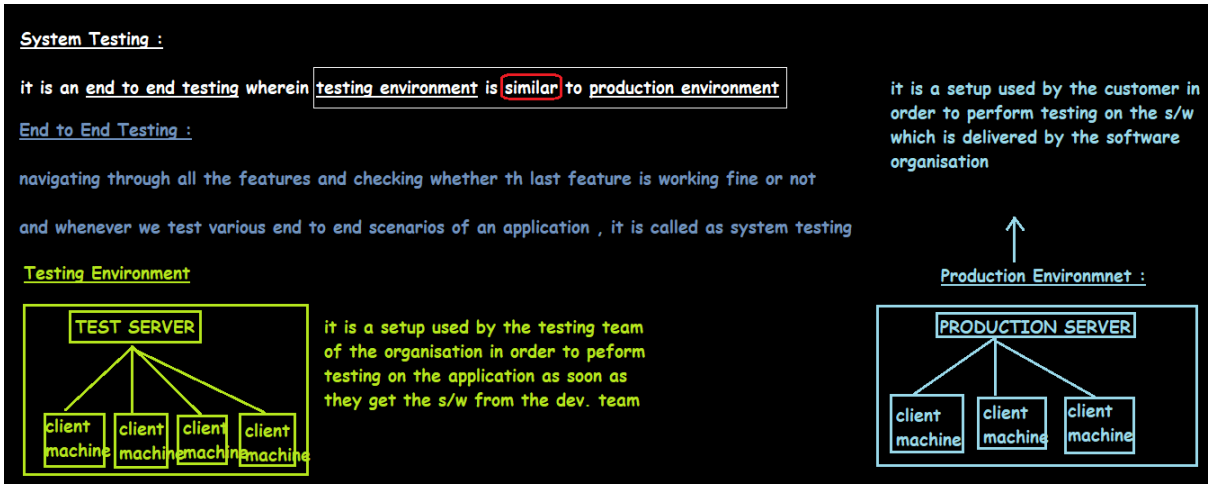
SYSTEM TESTING

It is end-to-end testing wherein testing environment is similar to the production environment.

End – to – end testing

Here, we navigate through all the features of the software and test if the end business / end feature works. We just test the end feature and don't check for data flow or do functional testing and all.

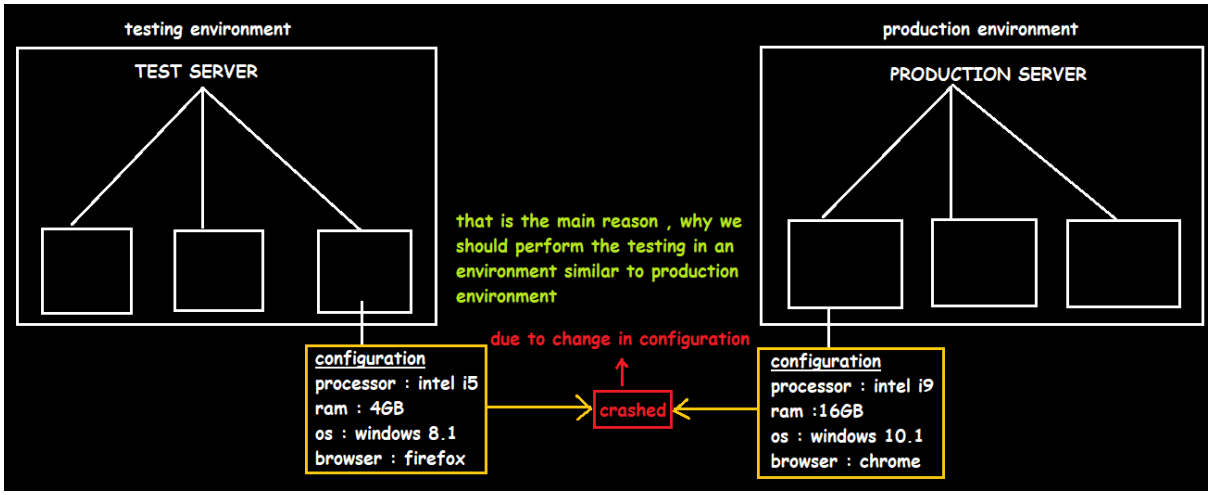




Thus, the actual definition of End-to-End testing can be given as – Take all possible end-to-end business flows and check whether in the software all the scenarios are working or not. If it is working, then the product is ready to be launched.

Why Testing Environment should be similar to Production Environment?

If the testing and production environment are not same chances are there the software may crash due to change in configuration at customer's place while performing the testing which is not a good thing at all. So, we should test the application in an environment similar to production environment.



When we do System Testing?

1. Minimum bunch of features must be ready
2. All the functionality and integration scenarios are working fine.
3. Testing environment is similar to production environment.

We say that the product is ready for release when,

a) All the features requested by customer are ready

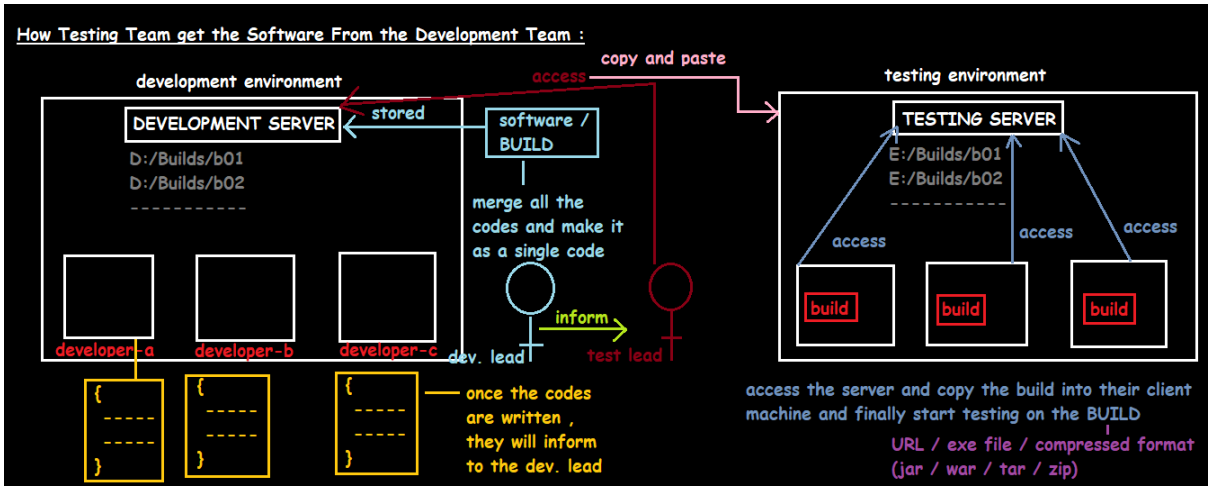
b) When all the functionality, integration and end-to-end scenarios are working fine

c) When there are no critical bugs. Bugs are there, but all are minor and less number of bugs

d) Product should be tested in an environment similar to production environment

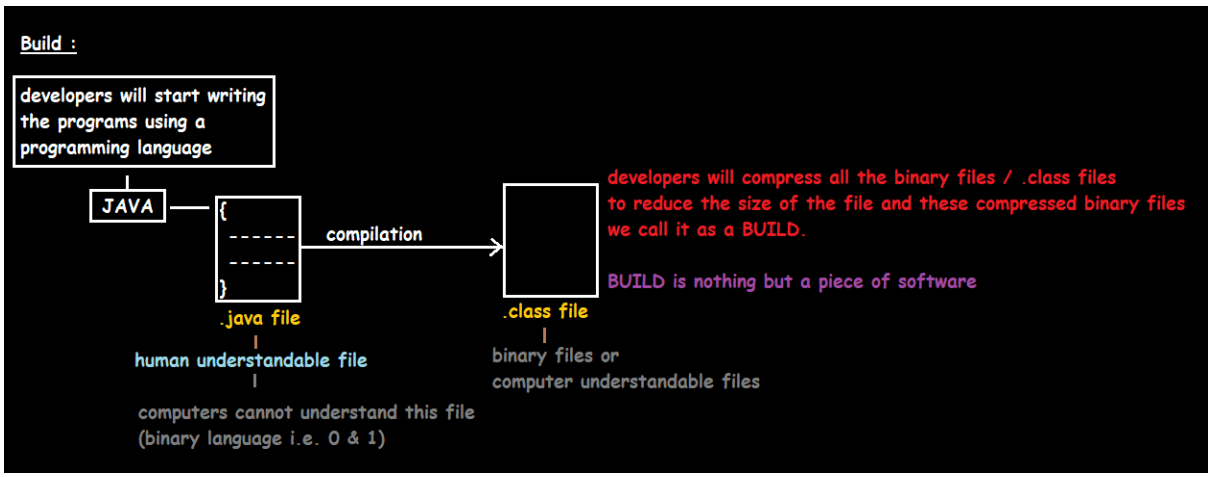
e) Whenever deadline is met

How the testing team gets the software from the development team?

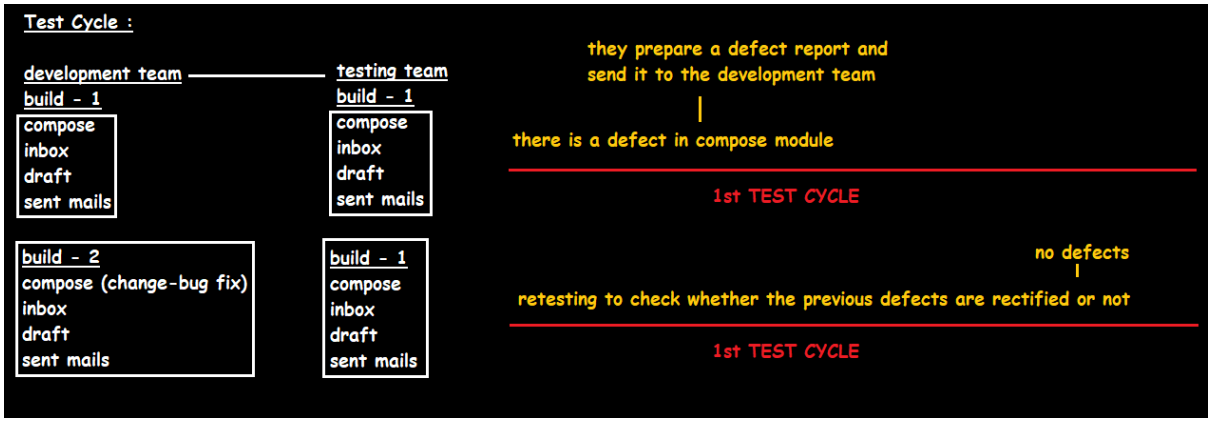


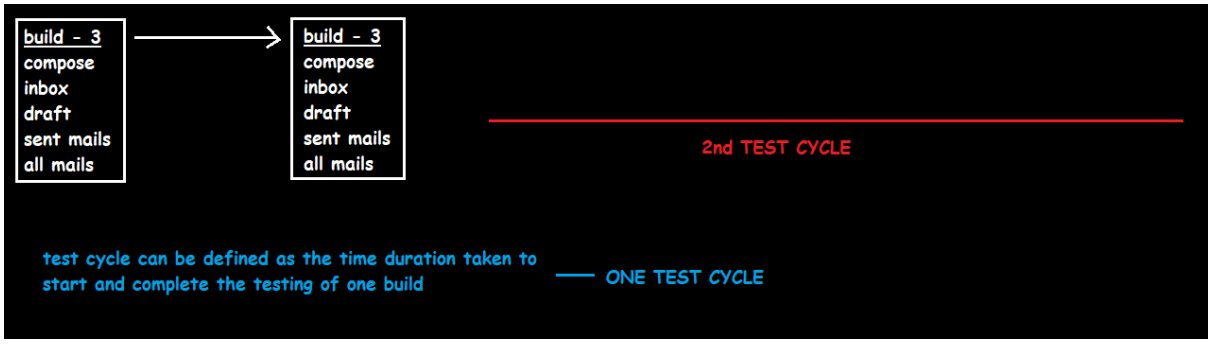
Build – Build is a piece of software which is copied, unzipped and installed at the testing server.

All the programs will be compiled and then compressed (compressed file should be in zip, war, tar, jar, exe, URL) format and compressed file is called build which is copied and pasted in the test environment, installed and we start testing the software.



Test cycle – Time duration taken to start and complete the testing of 1 build is called 1 Test Cycle.





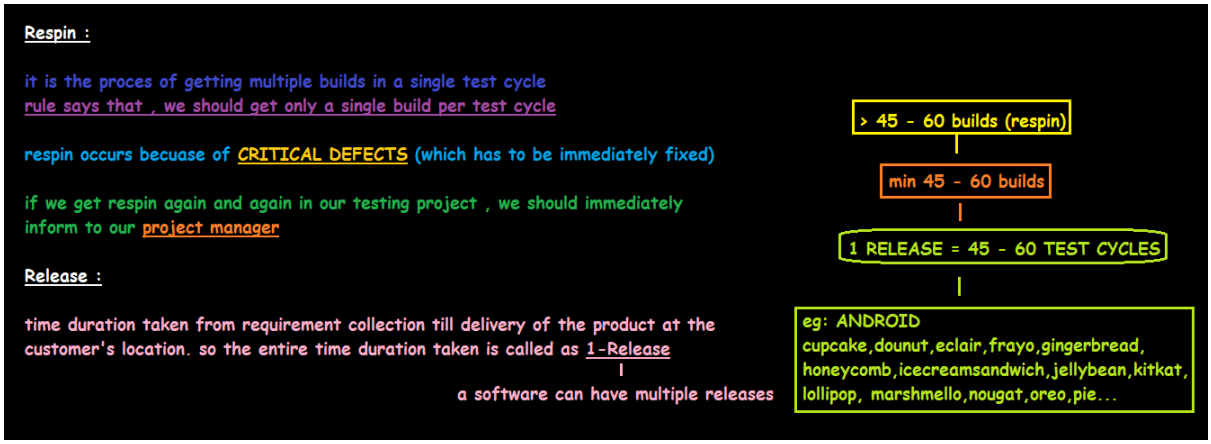
Respin- It is the process of getting more than one builds in a single test cycle.

We find respin – when the test engineer finds blocker defects / critical defects. For ex, if the login feature itself is not working in gmail, then the test engineer cannot continue with his testing and it has to be fixed immediately – thus respin comes into picture.

More number of respin in a cycle means the developer has not built the product properly.

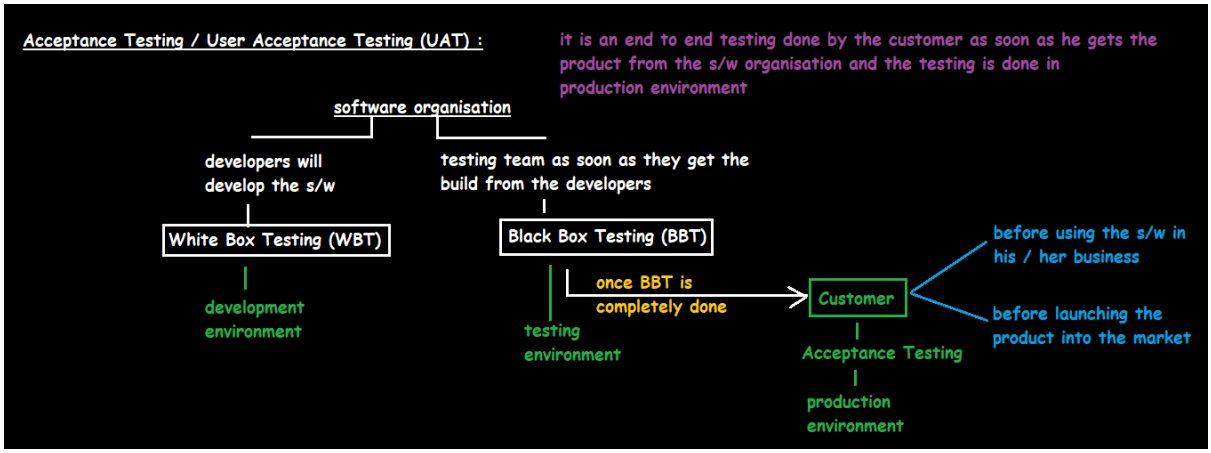
The entire period right from collecting requirements to delivering the s/w to the client is called as Release.

In interview, when they ask how many builds have u tested – optimum answer would be 26 – 30 builds.



ACCEPTANCE TESTING

It is an end to end testing done by the customer after receiving the product from the software organization and it is generally done in a production environment.



Why Customer Does Acceptance Testing?

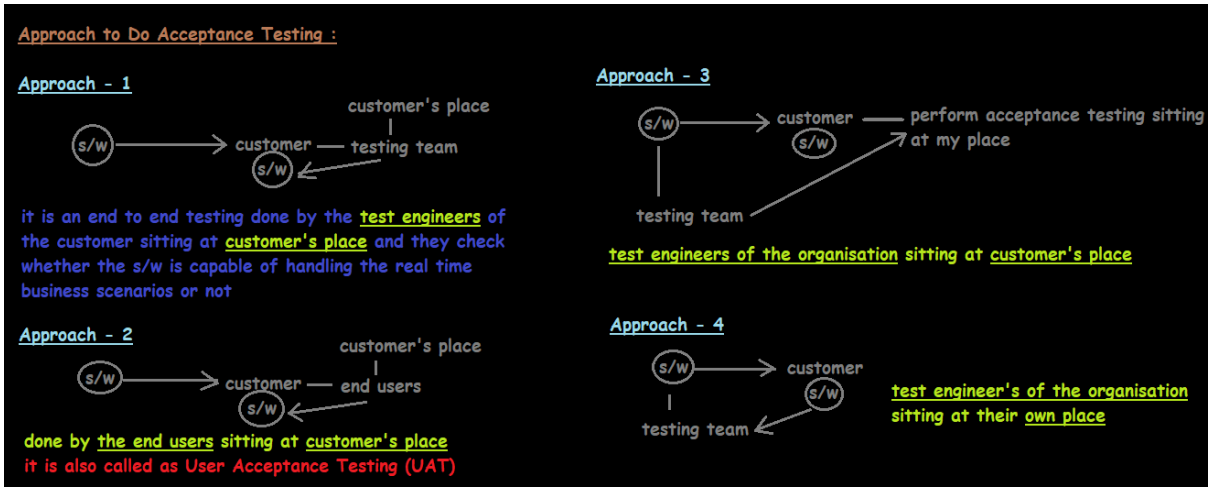
1. To check whether all end to end business scenarios are working fine or not and whether all the features requested by him are developed properly or not by the developers.
2. Under business pressure, chances are there software organisation might push the software with lots of defects. In order to find this, customer does acceptance testing.

We are getting more and more builds for Acceptance Testing means,

1. The product quality which is delivered to customers is not good. Development and testing both are not good.
2. After receiving the s/w, customer is getting more and more ideas, so he is asking for more and more changes

Acceptance testing can be done in four approaches.

1. It is an end to end testing done by end users who use the software for a particular period of time and they check whether the software is capable of handling the real time business scenarios or not.
2. It is an end to end testing done by test engineers of the customer sitting at the customers place wherein they check whether the software is capable of handling the real time business scenarios or not.
3. It is an end to end testing done by test engineers of the organisation sitting at the customers place wherein they check whether the software is capable of handling the real time business scenarios or not.
4. It is an end to end testing done by test engineers of the organisation in their own organisation wherein they check whether the software is capable of handling the real time business scenarios or not.



Types of Acceptance Testing

1. Alpha Testing
2. Beta Testing

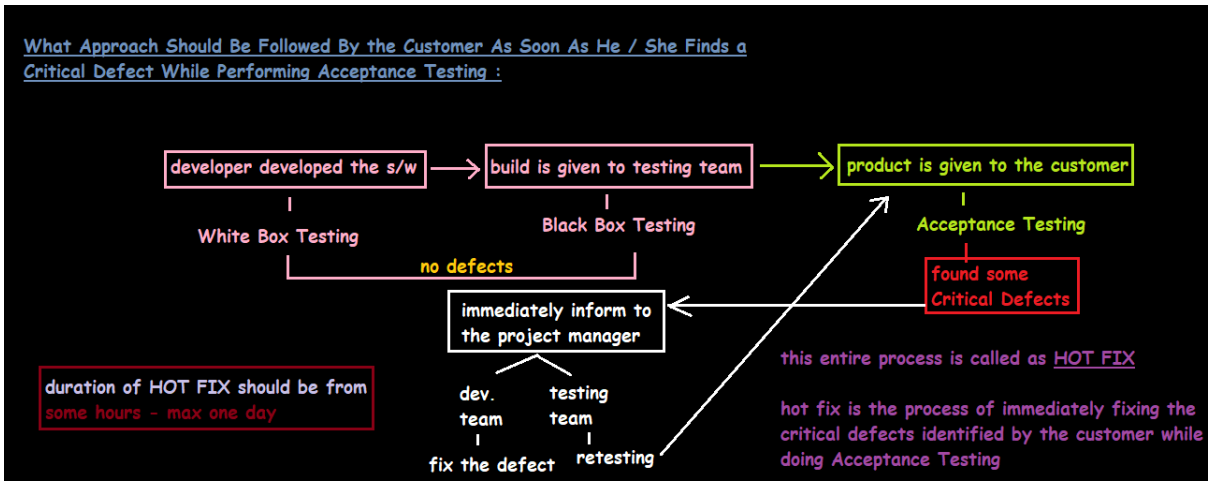
Alpha testing is a type of acceptance testing done at the **customers place** in production environment.

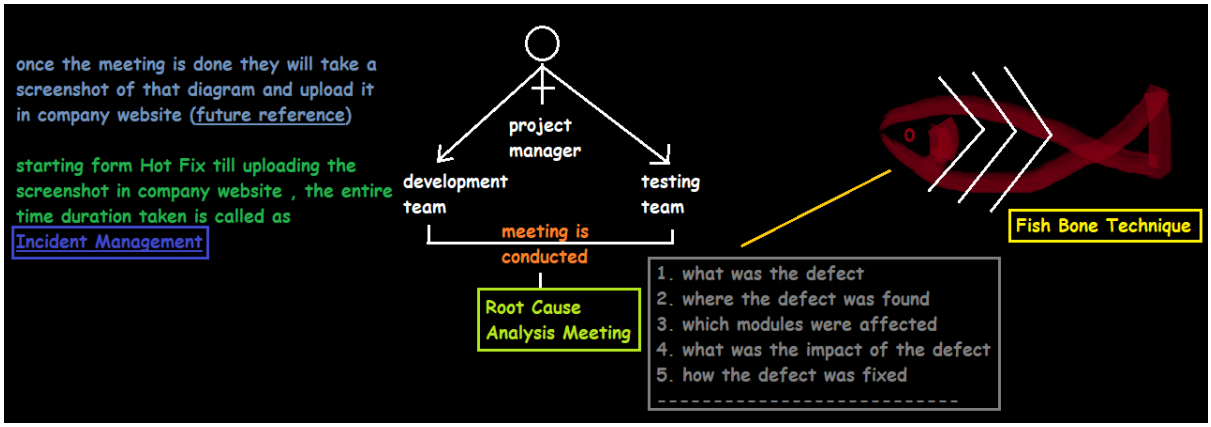
Beta testing is a type of acceptance testing done at the **software organisation** in production environment.

HOT FIX

It is the process of immediately fixing the critical defects identified by the customer while doing acceptance testing.

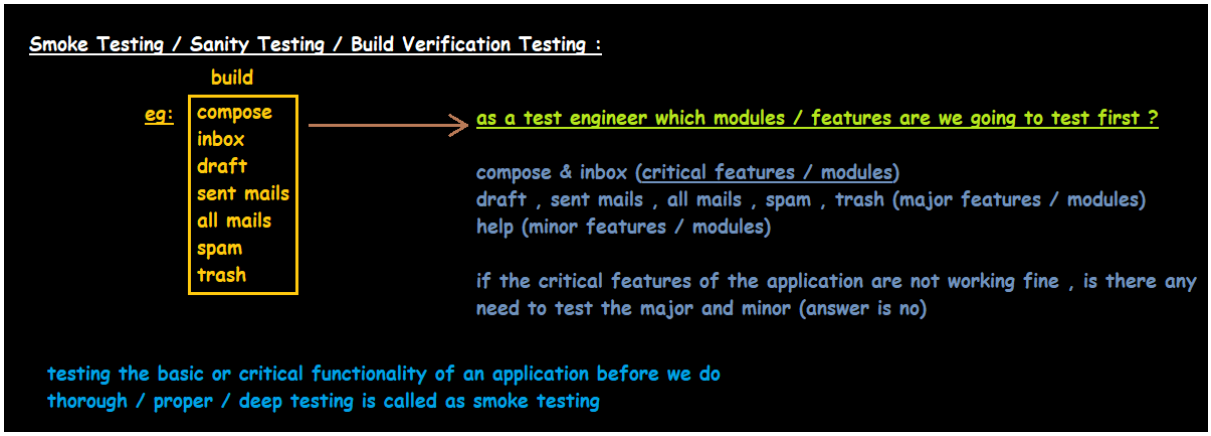
Duration of hot fix should be from some hours to max 1 day.





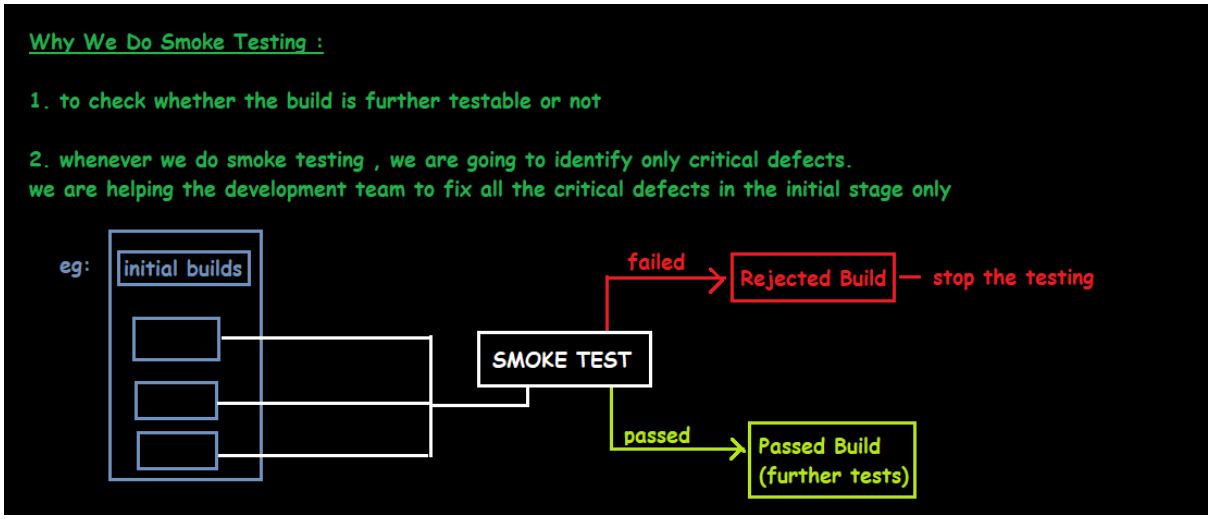
SMOKE TESTING or SANITY TESTING

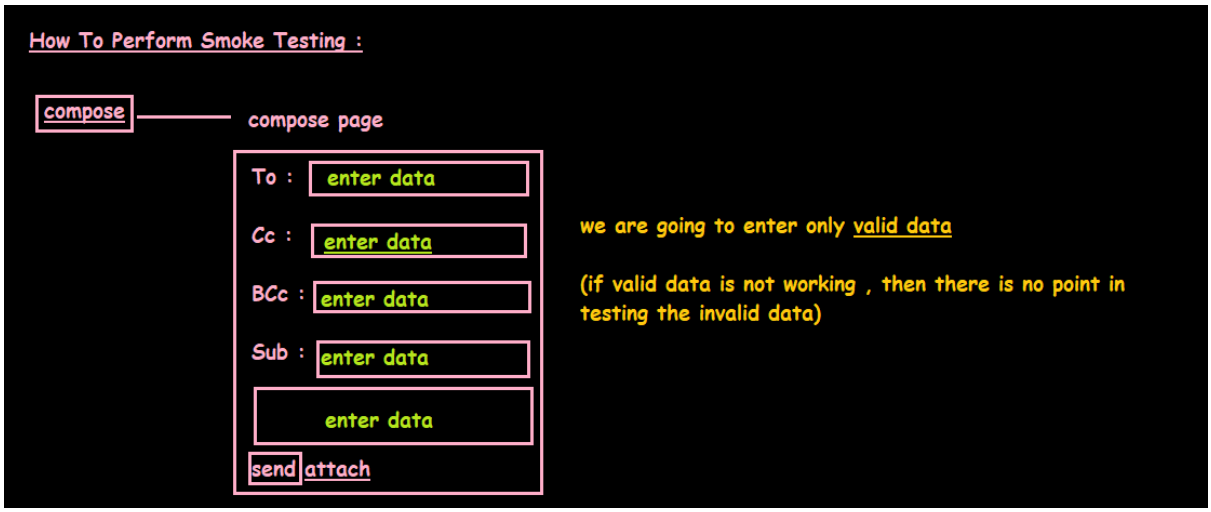
Testing the basic or critical features of an application before doing thorough testing or rigorous testing is called as smoke testing.

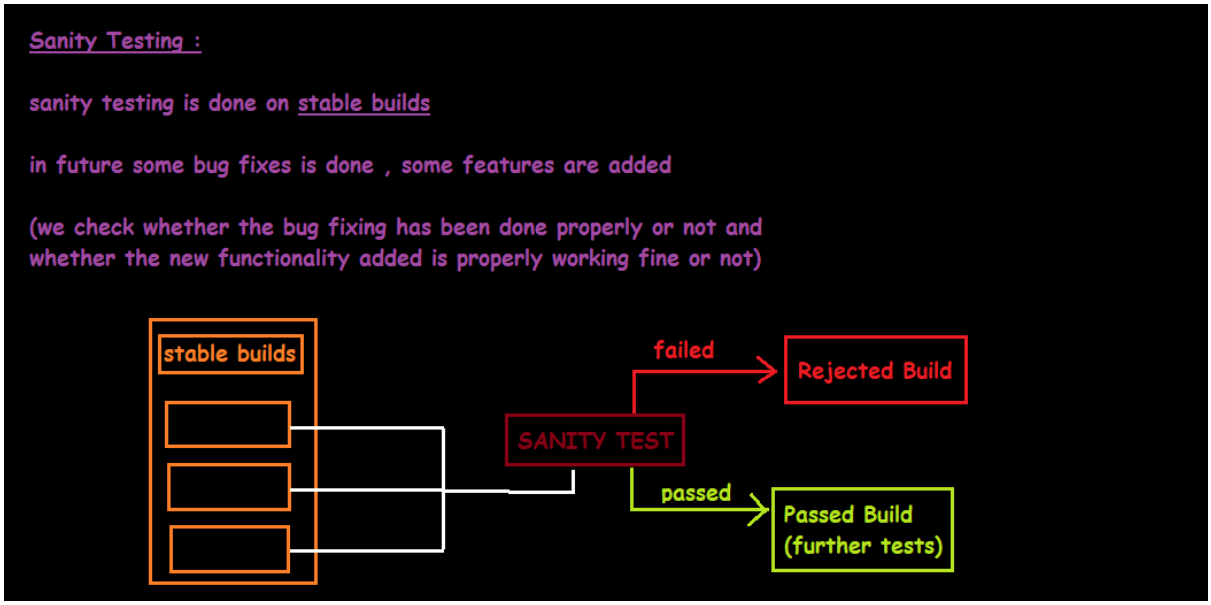


Why we Do Smoke Testing

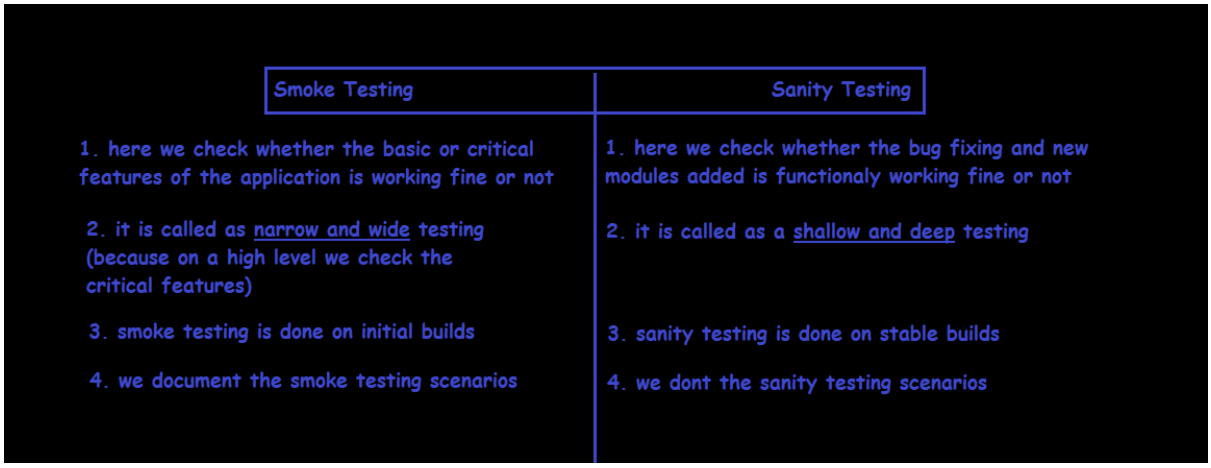
1. To check whether the product is further testable or not.
2. To help the developers fix the critical defects at an early stage.







Difference between Smoke & Sanity Testing



When we Do Smoke Testing

Whenever a new build comes in, we always start with smoke testing, because for every new build – there might be some changes which might have broken a major feature (fixing the bug or adding a new feature could have affected a major portion of the original software). In smoke testing, we do only positive testing – i.e., we enter only valid data and not invalid data.

Developers while doing White Box Testing, they do Smoke testing wherein first they check whether the critical lines of code are working fine or not. Then they check the remaining lines of code (major and minor).

When We Do Smoke Testing :

1. as soon as testing team they get the build from the development team , they do smoke testing

can the developers perform smoke testing :

2. while doing WBT , they first check whether all the critical lines of code are working fine or not

Note :

both development team and QA team can perform smoke testing
but sanity is generally performed by only the testing team

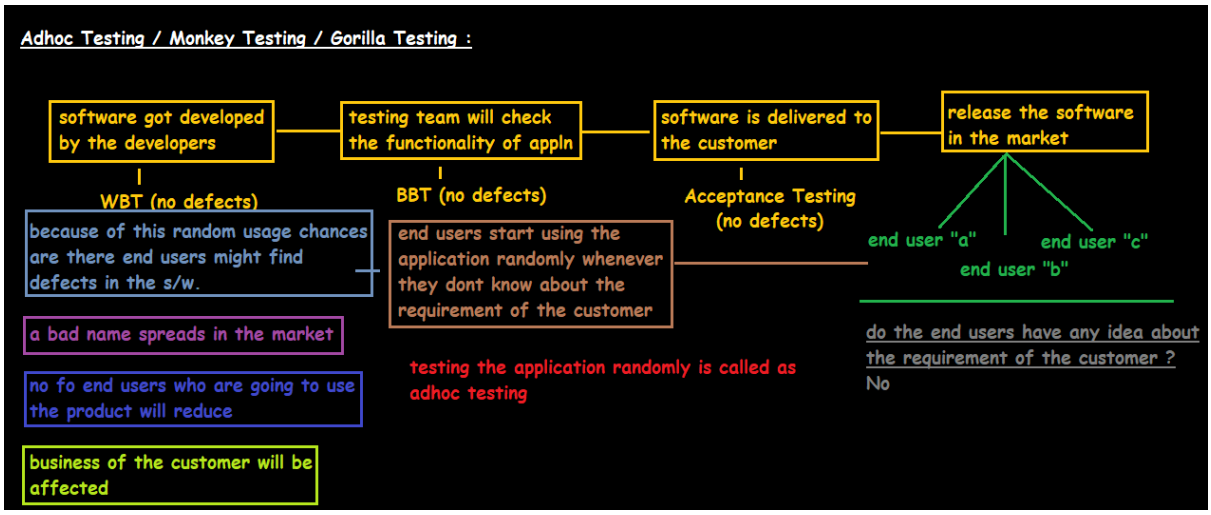
Important Points to Remember

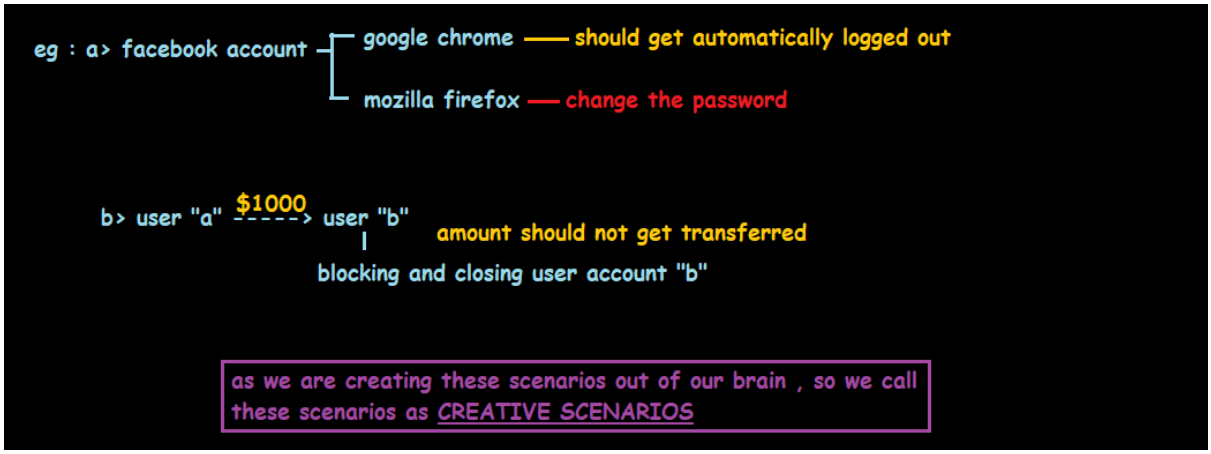
Note :

1. smoke testing is a kind of positive testing (because we check the functionality by giving positive or valid data)
2. in smoke testing we check only the critical features , we donot test the major and minor features because there is less time
3. in smoke testing , we execute the critical functionality , critical integration and critical system scenarios
4. in order to perform smoke testing , we should follow the requirement document (SRS)
5. smoke testing is also called as a health checkup of the build (Build Verification Testing)

AD – HOC Testing (also called Monkey Testing / Gorilla Testing)

Testing the application randomly is called Ad-hoc testing.





Why we do Ad-Hoc Testing

1) End-users use the application randomly and he may see a defect, but professional TE uses the application systematically so he may not find the same defect. In order to avoid this scenario, TE should go and then test the application randomly (i.e. behave like and end-user and test)
2) Development team looks at the requirements and build the product. Testing Team also look at the requirements and do the testing. By this method, Testing Team may not catch many bugs. They think everything works fine. In order to avoid this, we do random testing behaving like end-users
3) Ad-hoc is a testing where we don't follow the requirements (we just randomly check the application). Since we don't follow requirements, we don't write test cases.

When to do Ad-Hoc testing?

- Whenever we are free, we do Ad-hoc testing.
- After testing as per requirements, then we start with ad-hoc testing
- In the early stage of product development we should not think of doing Adhoc Testing

In early stages of product development, doing smoke testing fetches more number of bugs. But, in later stages of product development, if you do smoke testing – the number of bugs that you are going to catch in smoke testing will be very less. Thus, gradually the effort spent on smoke testing is less.

Why We Do Adhoc Testing :

1. after getting the product from the customer , end users might use the product randomly and they might find defects and a bad name will spread in the market and no of end users who are going to use the product will reduce and it is going to affect the business of the customer

2. if we see the requirement document and test the product , no of defects we are going to catch will be less , so inorder to find out more and more no of defects , we do adhoc testing

When We Do Adhoc Testing :

1. whenever testing team they get free time , they should do adhoc testing

2. in the early stage of product development no of defects we are going to catch will be more. so at that time we dont think about adhoc testing

3. 

NOTE:-

Ad-hoc testing is basically negative testing because we are testing against requirements

(Out of requirements)

Here, the objective is to somehow break the product and we don't follow the requirement document.

Note :

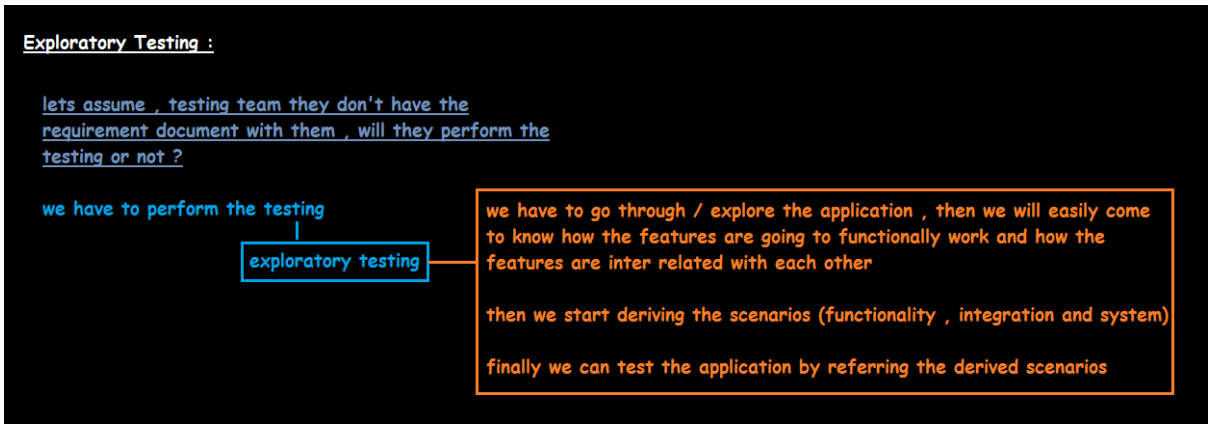
1. adhoc testing is a kind of negative testing (testing the application randomly)

2. to do adhoc testing , no need to follow the requirement document

Smoke Testing	Adhoc Testing
1. we test the critical features of an application whether it is working fine or not	1. we test the application randomly
2. as soon as we get the build from the development team	2. whenever we get free time , we do adhoc testing
3. smoke testing is a kind of positive testing	3. adhoc testing is a kind of negative testing
4. to do smoke testing we need the requirement document	4. to do adhoc testing no need to follow the requirement document

EXPLORATORY TESTING

Explore the application, understand the application and based on understanding we will easily come to know how all the features functionally works and how all the features are inter related to each other and then try to identify the scenarios (functionality , integration and system scenarios) and test the application based on the identified scenarios.

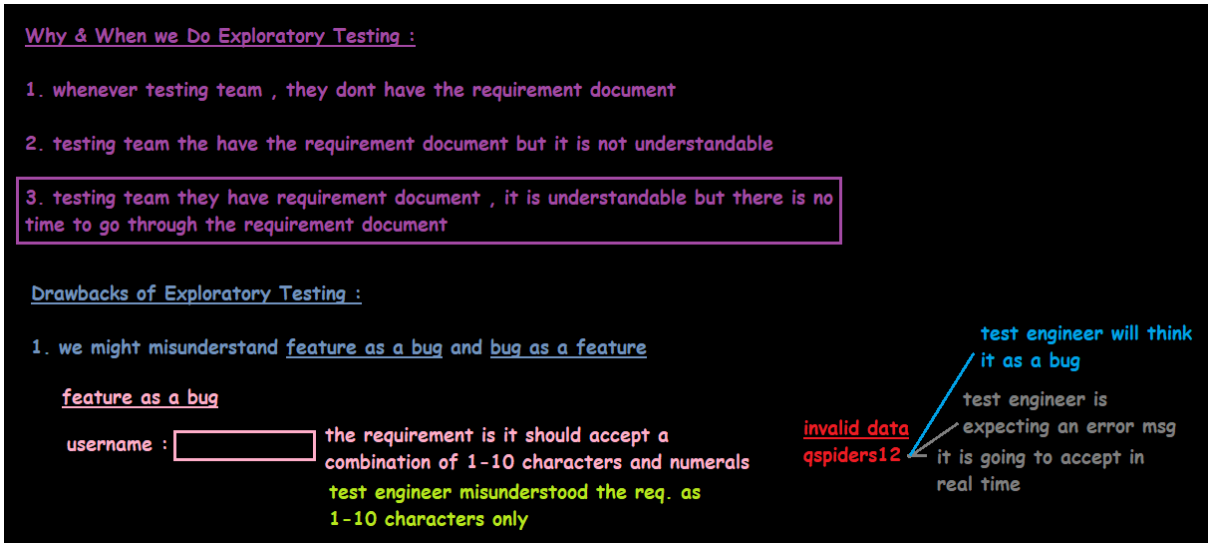


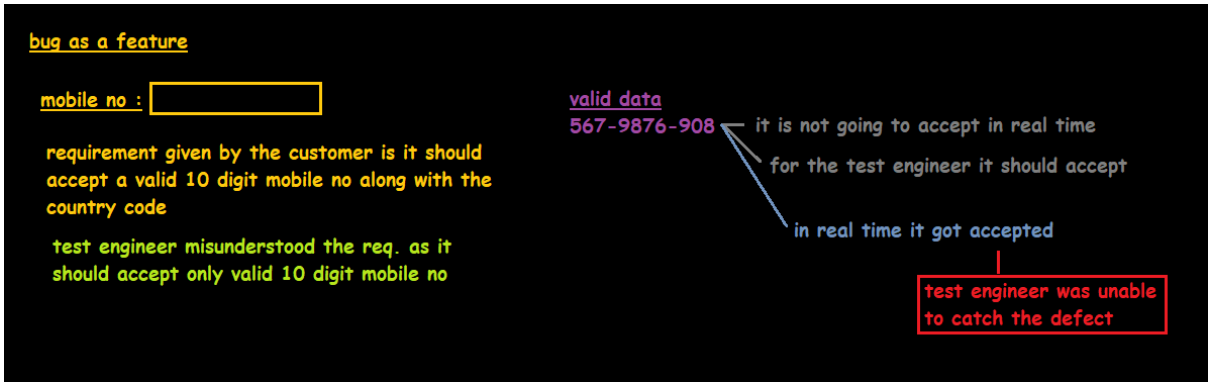
Why/When we do Exploratory Testing

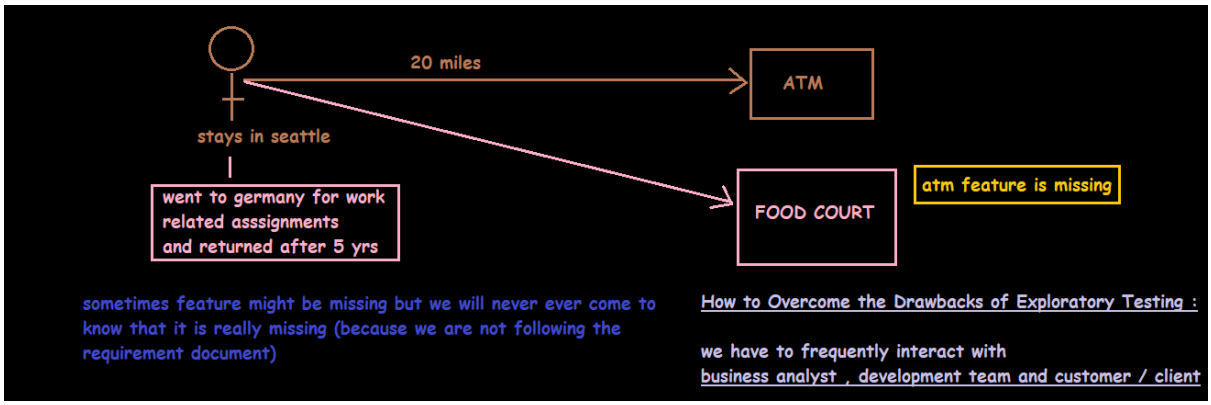
1) When there is no requirement document
2) Requirement is there but it is not understandable.
3) Requirement is there, it is understandable but there is no time to go through the requirement.

Drawbacks of Exploratory Testing:

1. We might misunderstand a feature as a bug and bug as a feature.
2. Sometimes feature is only missing but we never come to know that it is really missing.







How to Overcome the Drawbacks:

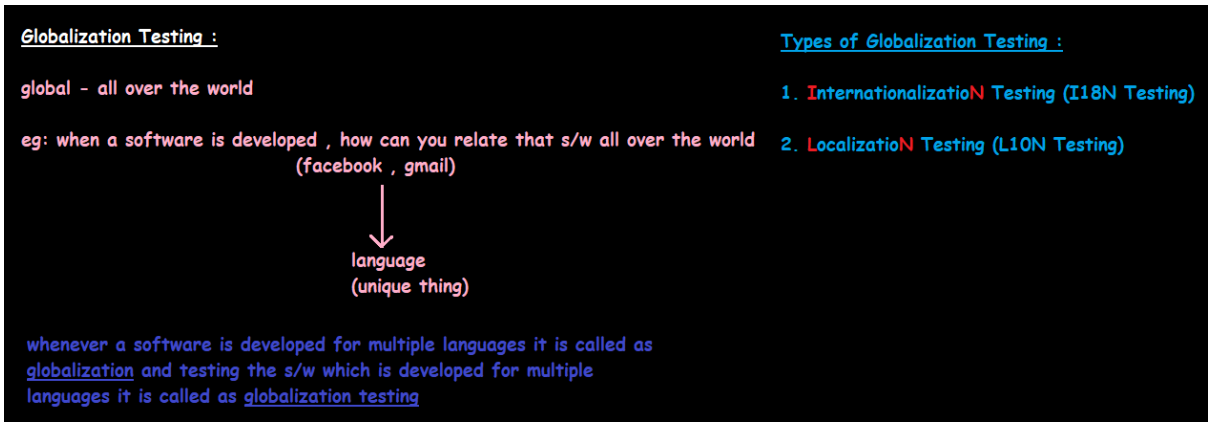
1. Communicate constantly with the client or Business Analyst or the Development team.

GLOBALIZATION TESTING

Developing a software for multiple languages is called as Globalization and testing the software which is developed for multiple languages is called as Globalization Testing.

TYPES OF GLOBALIZATION TESTING:

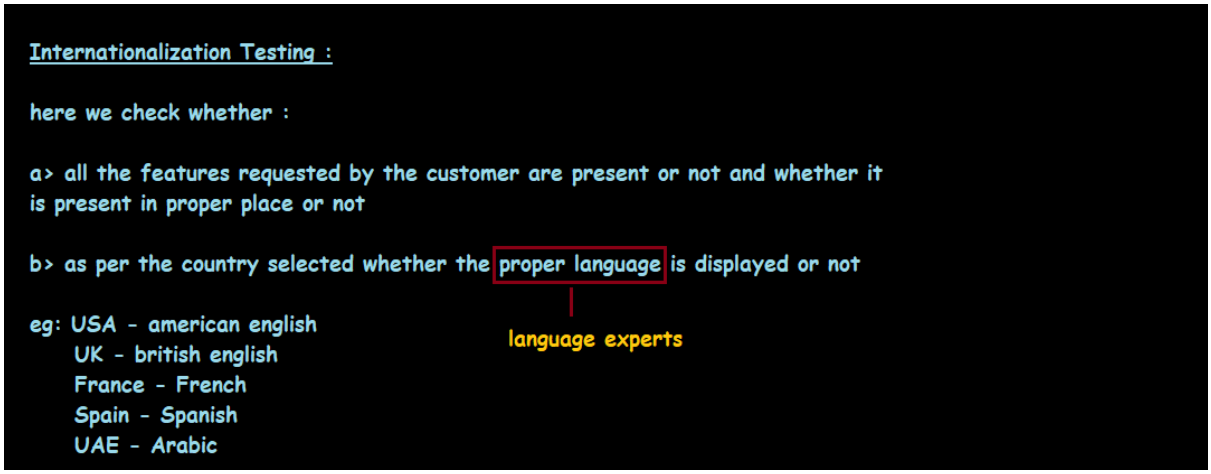
1. INTERNATIONALIZATION (I18N) TESTING
2. LOCALIZATION (L10N) TESTING



INTERNATIONALIZATION (I18N) TESTING:

Here we check whether all the features requested by the customer are present or not and whether it is present in right place or not.

Also, we check whether as per the country selected, features are displaying in proper language or not.

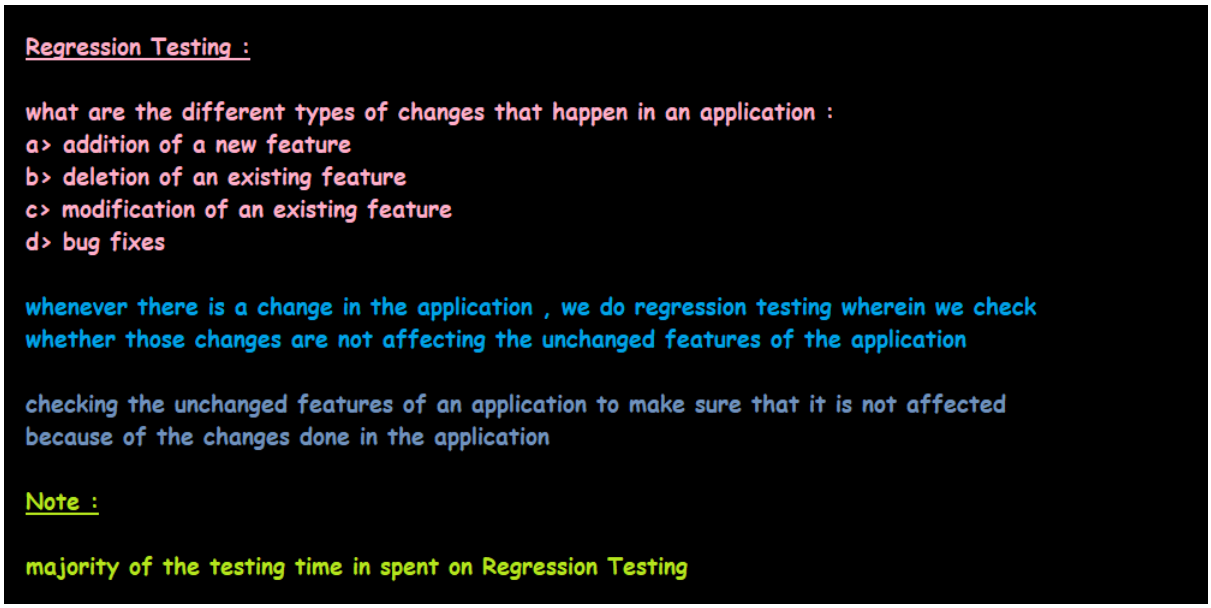


LOCALIZATION (L10N) TESTING:

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

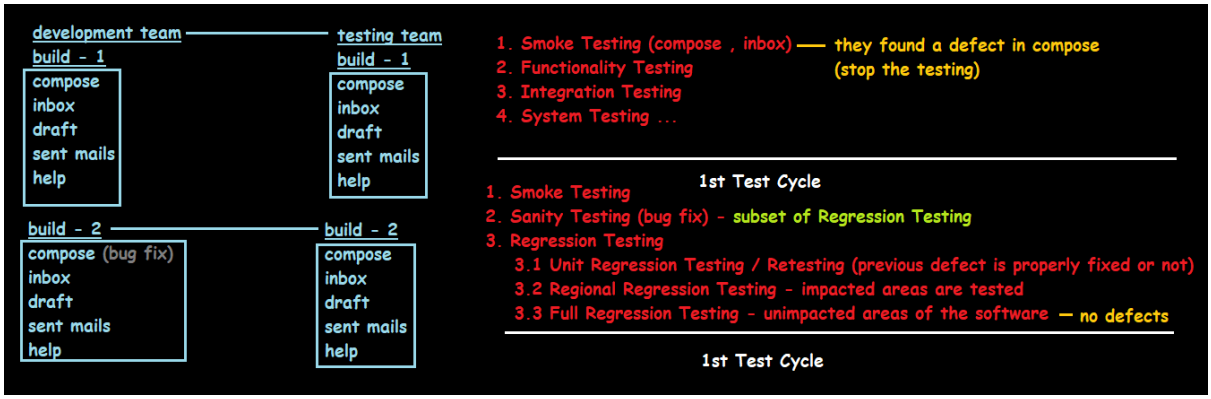
REGRESSION TESTING :

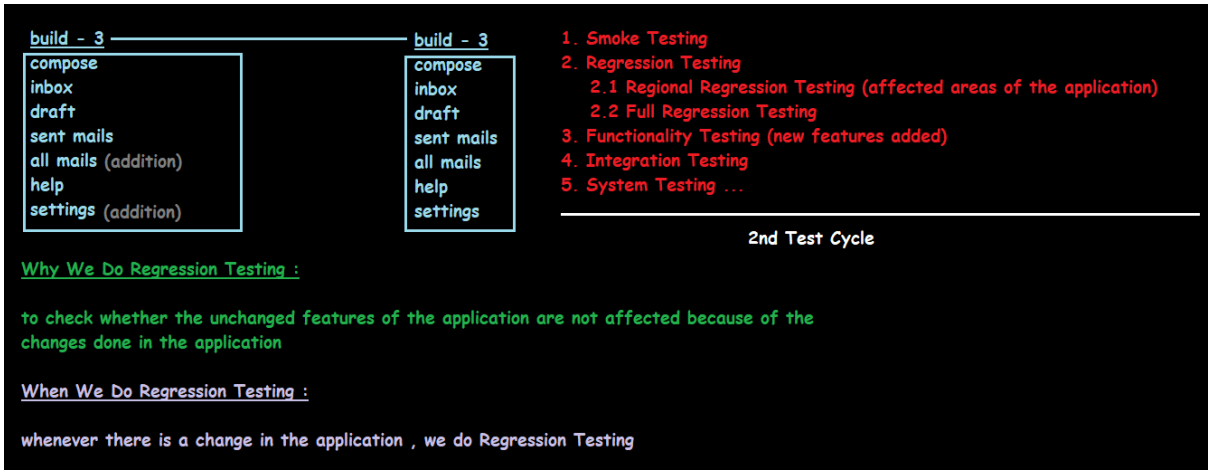
Testing the unchanged features to make sure that it is not broken because of the changes (changes means – addition, modification, deletion or defect fixing) is called regression testing.



When the development team gives a build, chances are there they would have done some changes. That change might affect unchanged features. So, Testing the unchanged features to make sure that it is not broken because of the changes is called Regression Testing.

Majority of time spent in testing is on regression testing.





Based on changes, we should do different types of regression testing,

- Unit Regression Testing
- Regional Regression Testing
- Full Regression Testing

a) Unit Regression Testing (URT)

Testing the changes i.e. nothing but the bug fixes is called URT.

In Build B01, a bug is found and a report is sent to the developer. The developer fixes the bug and also sends along some new features developed in the 2nd build B02. The TE tests only if the bug is fixed.

Testing only the modified features is called Unit Regression Testing

b) Regional Regression Testing (RRT)

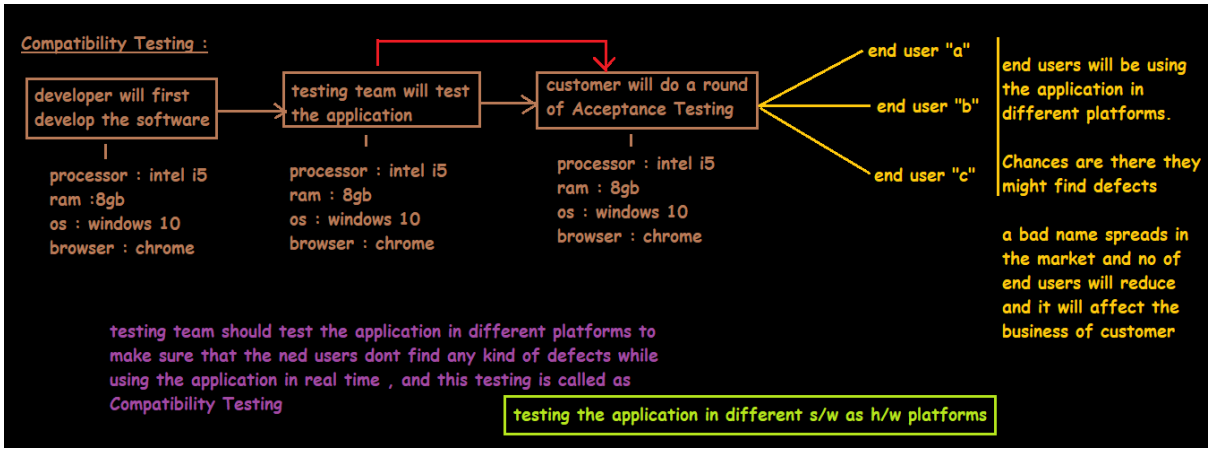
Testing the changes and impact regions is called Regional Regression Testing.

c) Full Regression Testing

Thus, testing the changes and all the remaining features is called Full Regression Testing.

COMPATIBILITY TESTING

Testing the functionality of an application in different software and hardware environment is called Compatibility testing.

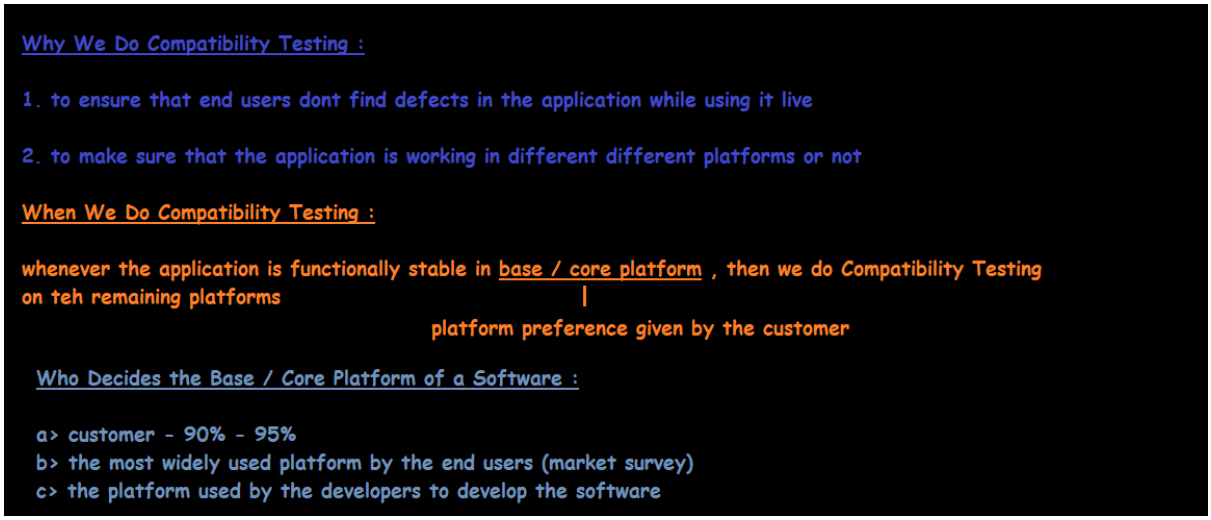


Why We Do Compatibility Testing

1. we might have developed the s/w in 1platform – chances are there that users might use it in different platforms – thus it could lead to defects and bugs – people may stop using the s/w – thus business will be affected and hence we do Compatibility Testing.
2. To check whether the application is functionally working fine in different platforms or not.

When we Do Compatibility Testing

Whenever the software is functionally stable in base or core platform.

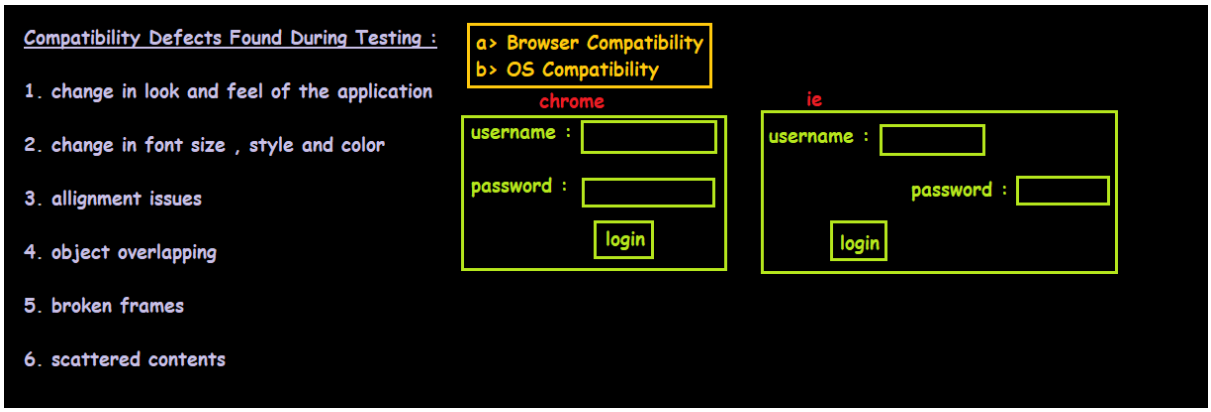


How to decide platform as Base Platform

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

The various Compatibility bugs are,

- Scattered content
- Alignment issues
- Broken frames
- Change in look and feel of the application
- Object overlapping
- Change in font size, style and colour



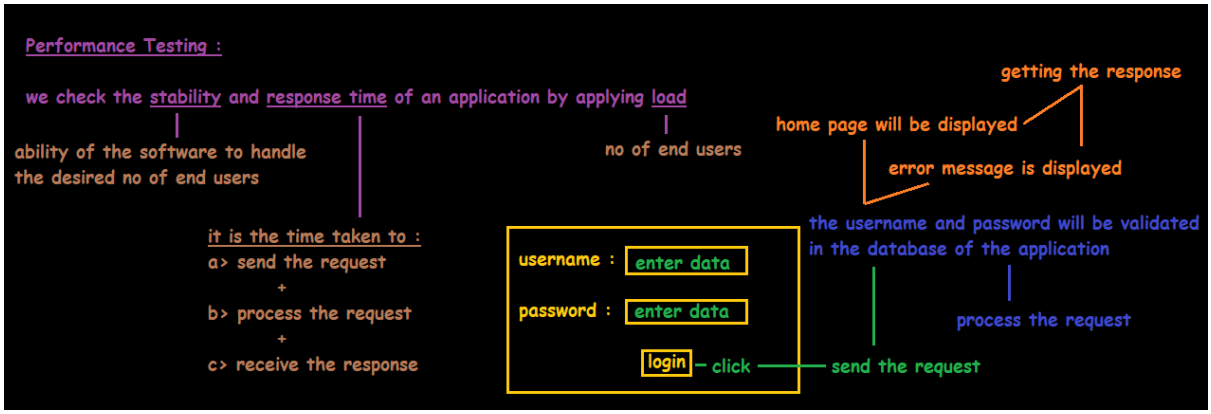
PERFORMANCE TESTING

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

Stability in the sense the ability to withstand the load of desired no of users.

Response time is the time taken to send the request + time take to run the program + time taken to send the response.

Load is the no of users trying to access the server at a time.



TYPES OF PERFORMANCE TESTING

1. Load Testing
2. Stress Testing
3. Volume Testing
4. Soak Testing

LOAD TESTING

Testing the stability and response time of an application by applying load which is less than or equal to the desired no of users.

STRESS TESTING

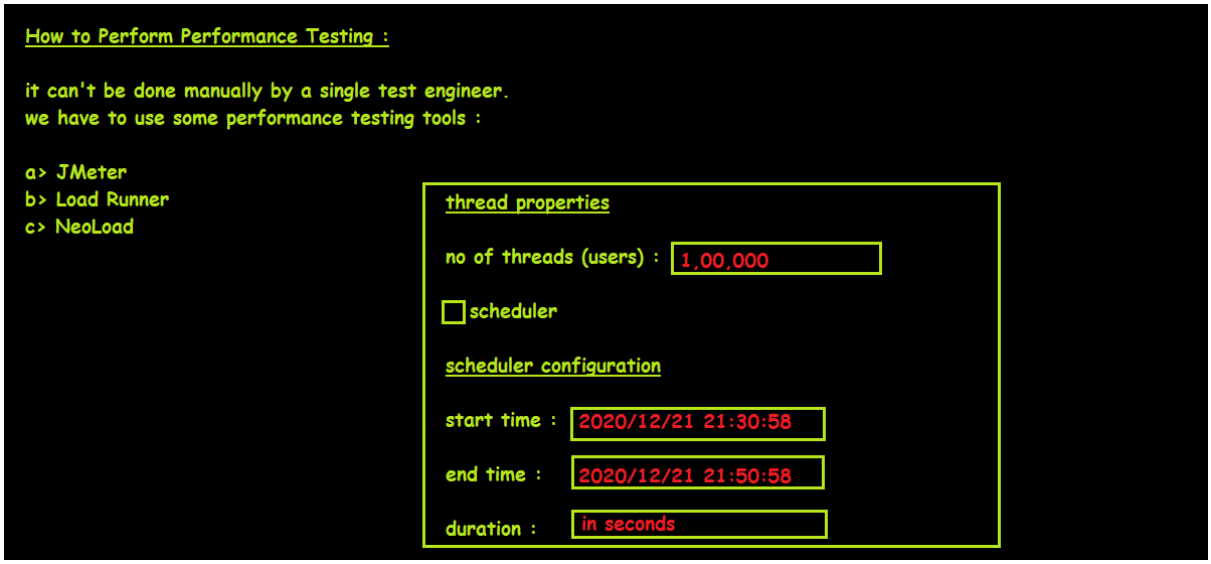
Testing the stability and response time of an application by applying load more than the desired no of users.

VOLUME TESTING

Testing the stability and response time of an application by transferring large volume of data through it.

SOAK TESTING

Testing the stability and response time of an application by applying load continuously for a particular period of time.



USABILITY TESTING:

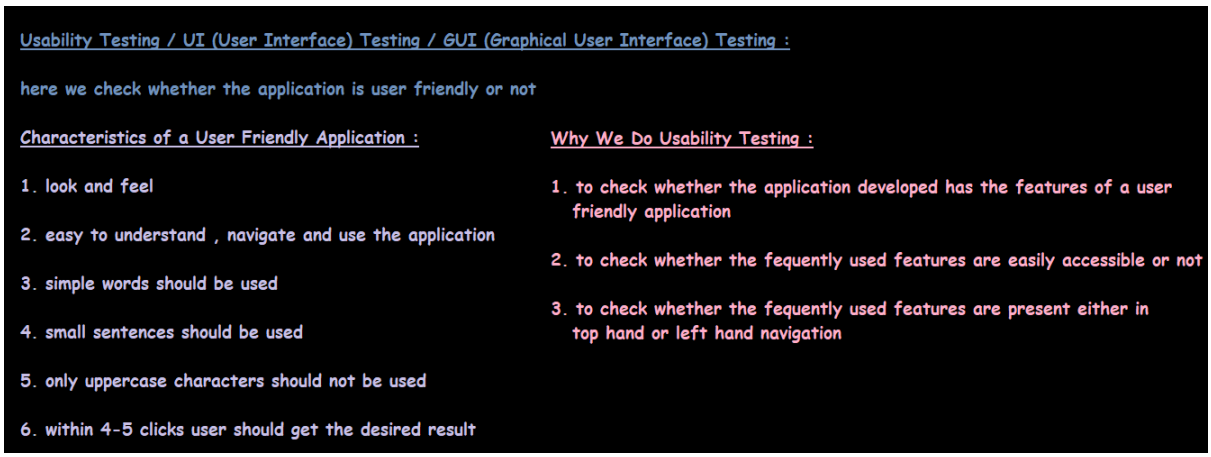
Testing the user friendliness of an application, we call it as Usability Testing.

Why we do Usability Testing

1. To check whether frequently used features are easily accessible or not.
2. To check whether frequently used features are displayed either in left or top navigation bar.

OS's GUI standards/Characteristics of an USER FRIENDLY software

1. Look and feel good.
2. Easy to use
3. Easy to understand
4. Easy to navigate
5. It should take very less time (within 3 clicks) to reach what user wants
6. Small sentences should be there
7. Simple words should be there.
8. Lower case letters should be used



RELIABILITY TESTING:

Testing the functionality of an application continuously for a particular period of time.

For ex – let us consider our cellphones / mobiles. The s/w 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 any call or message – we will be unable to press the call button and the phone hangs up. Thus we make use of a clean-up software. When we switch off the phone and switch it on again, all the objects in the RAM gets deleted.

RECOVERY TESTING:

Testing the application to check how well it recovers from crashes or disasters

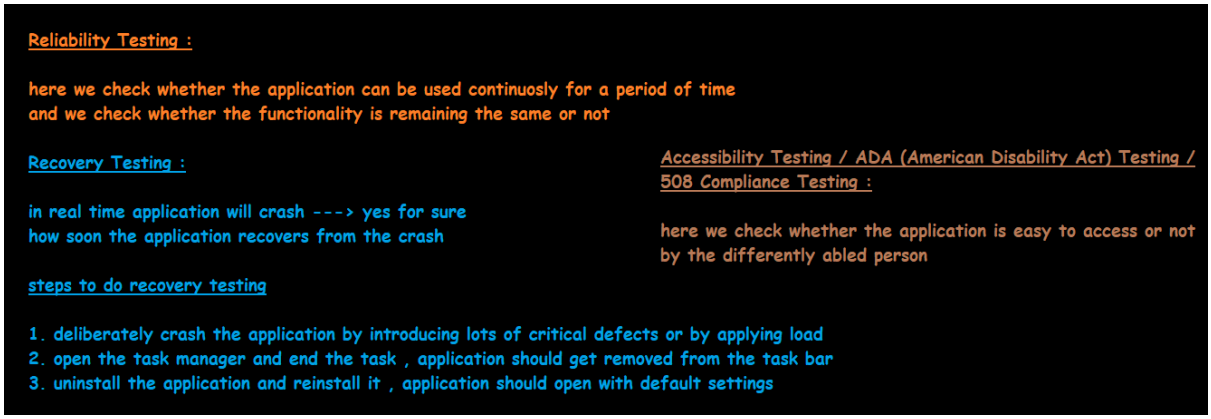
The steps involved in Recovery Testing are,

1. Introduce defect and crash the application- By experience after few months of experience on working the project, we can get to know how and when the s/w can and will crash.
2. Make sure that it is crashed fully.
   a. Whenever application crashed, that application disappears from the screen.
   b. Press CTRL+ALT+DEL, open task manager, that crashed application process should not be there.
3. Uninstall the application which is crashed.
4. Reinstall it once again.
5. Double click and open the application, application should open with default settings.

ACCESSIBILITY TESTING / ADA TESTING (American Disability Act) /

508 COMPLIANCE TESTING:

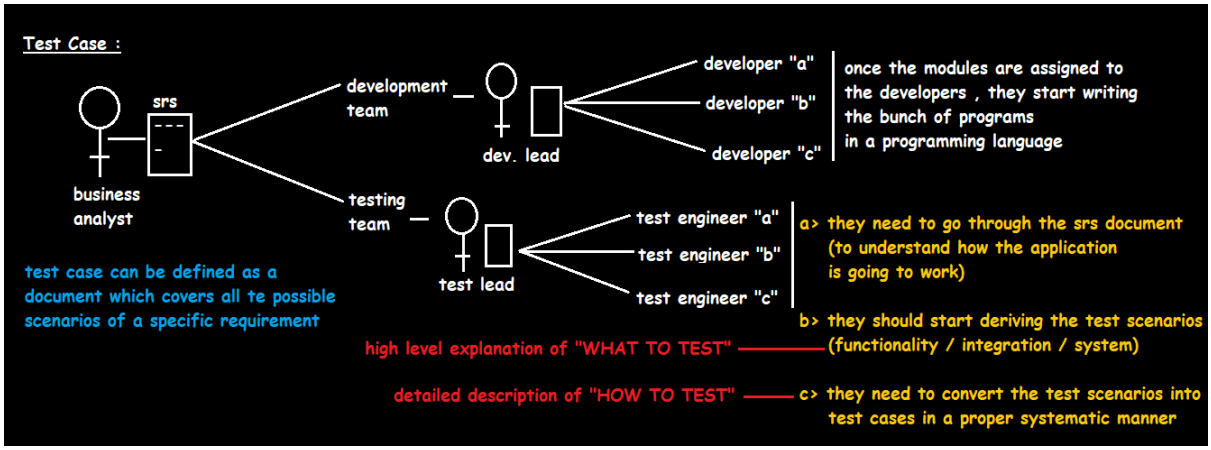
Here, we check whether the application can be easily accessed or can be easily used or not by physically challenged person.



TEST CASE

Test case is a document which covers all possible scenarios to test all the feature(s).

It is a set of input parameters for which the s/w will be tested.



Why we write test cases?

1. To have better test coverage – cover all possible scenarios and document it, so that we need not remember all the scenarios

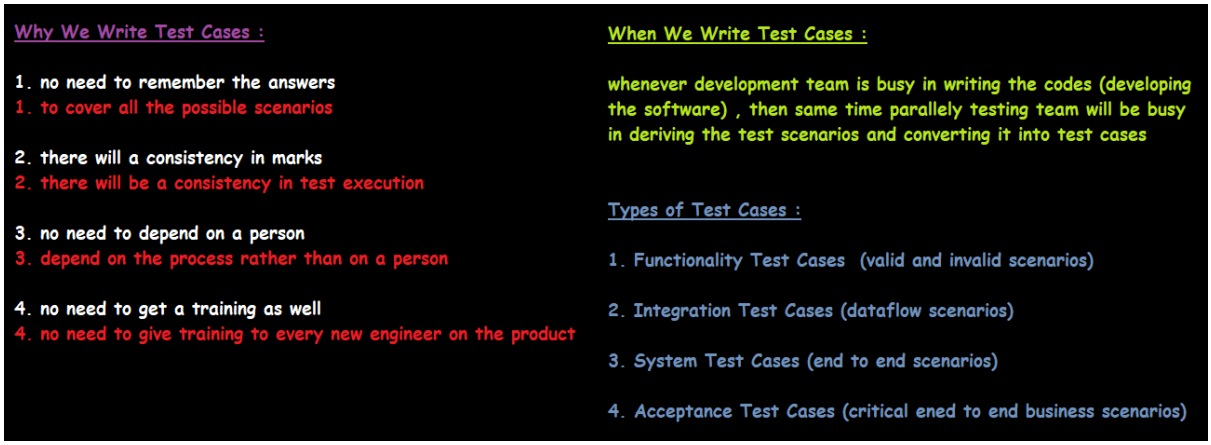
2. To have consistency in test case execution – seeing the test case and testing the product

3. To avoid training every new engineer on the product – when an 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.

4. To depend on process rather than on a person

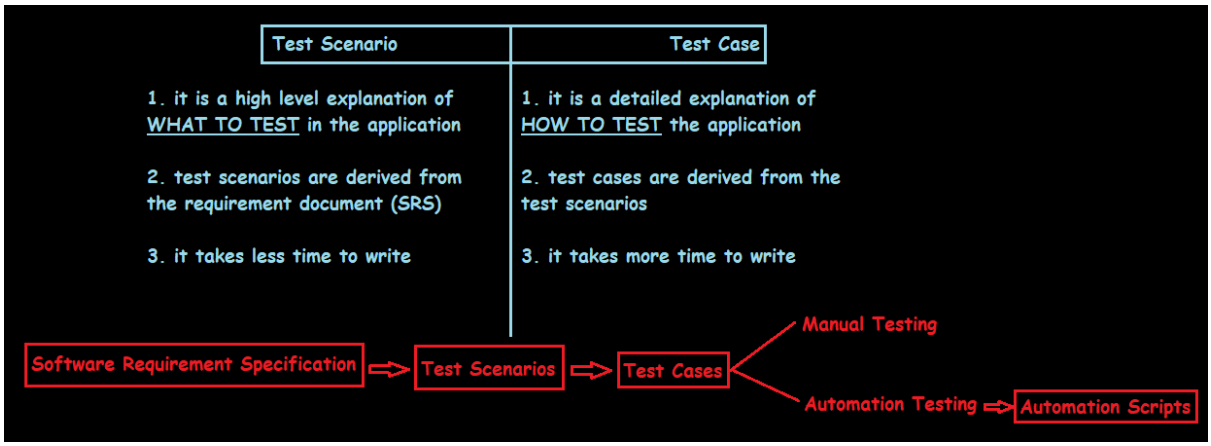
When do we write test cases?

Customer gives requirements – developer start developing the product– during this time, testing team start writing test cases by referring the test scenarios.



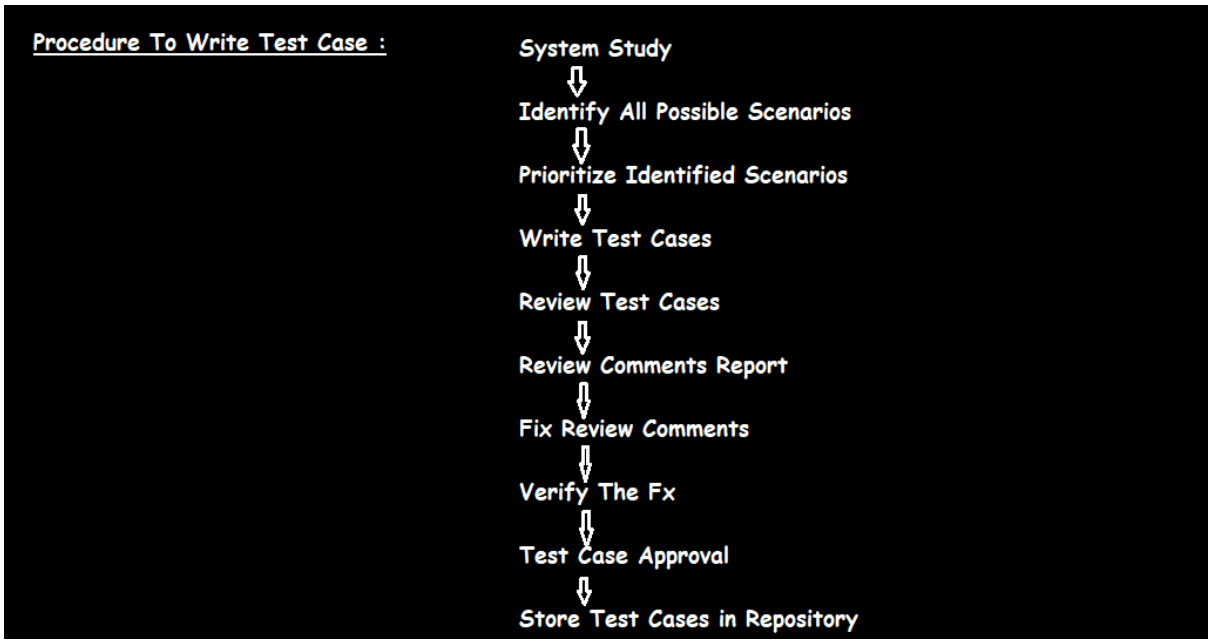
Difference between Test Scenario and Test Case

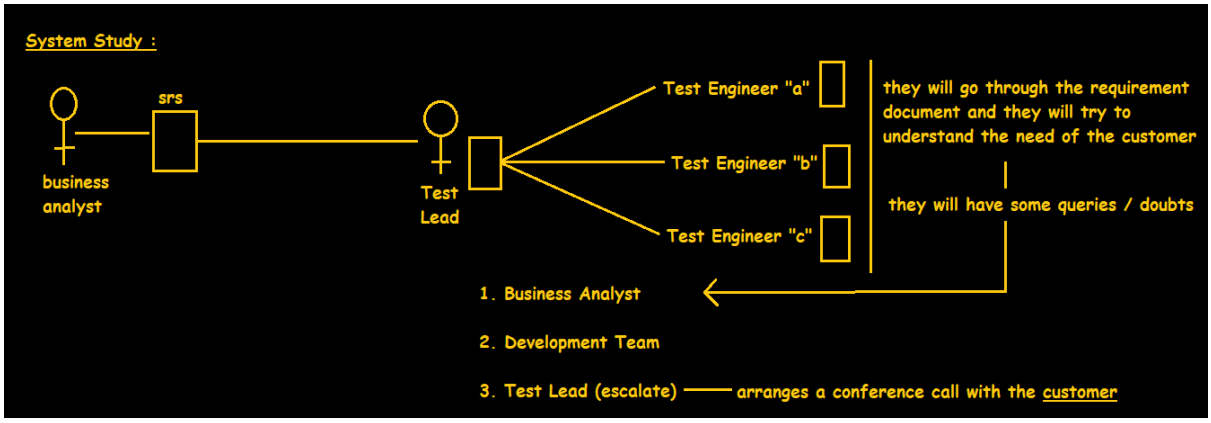
1. Test scenario is a **one line explanation** of "**what to test**" in the application whereas Test case is a **detailed explanation** of "**how to test**" the application.
2. Test Scenarios are derived from the requirement document whereas Test Cases are derived from the Test Scenarios.
3. Test Scenarios takes less time to write and test cases take more time to write.



Test Case Template:

1	Test Case Name						
2	Module Name						
3	Severity						
4	Test Case Type						
5	Pre Condition						
6	Brief Description						
7							
8	Sl No	Action (Description)	Input / Test Data	Expected Result	Actual Result	Status	Comments
9							
10							
11							
12							
13							
14	Author's Name						
15	Reviewer's Name						
16	Approved By						
17	Approval Date						





Identify All The Possible Scenarios :

once the test engineers have understood the requirement of the customer ,
they start identifying all the possible scenarios

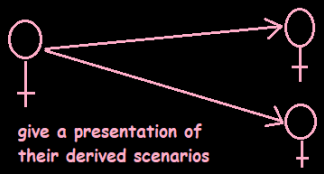
a. functionality scenarios (valid & invalid data)

b. integration scenarios (dataflow scenarios)

c. system scenarios (end to end scenarios)

once the scenrios are derived , then we should inform to our Test Lead

arranges a session and that session is called as a "BRAIN STORMING SESSION"



give a presentation of their derived scenarios

they might help you to identify your wrong , missing & repeated scenarios

Prioritize Identified Scenarios :

once the scenarios are derived , then the test engineers should prioritize all the identified scenarios on the basis of the impact those scenarios are having on the business of the customer

priority is set as CRITICAL , MAJOR , MINOR

Write Test Cases :

once the test scenarios are derived and prioritized , then the test engineer's (author) start writing the test cases.

author will refer the scenarios and start convering the same into test cases and he / she will prepare functionality / integration / system test cases

Review Test Cases :

once the test cases are written , review is done to identify the mistakes present in author's test cases and review is done by another test engineer and we call him / her as reviewer

Why We review Test Cases :

1. to identify the wrong / missing / repeated scenarios

2. to check whether all the attributes (header , body and footer) are present or not

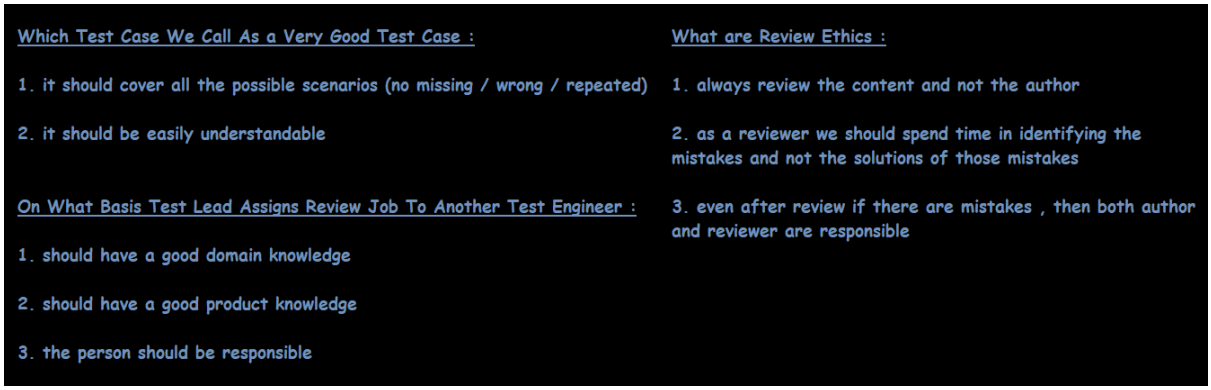
3. to check whether all the attributes contain relevant data or not

4. to check whether the test case is easily understandable or not

5. to check whether standard test case template is followed or not

Why we Review Test Cases?

1. To find missing scenarios, wrong scenarios & repeated scenarios.
2. To check whether 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.
3. To check whether all the attributes are covered or not.
4. To check whether all the attributes contains relevant data or not.
5. To check whether standard test case template is followed or not.



Which test case we call as a very good test case?

1. Which cover all possible scenarios.
2. Should be easily understandable.

On what Basis Test Lead assigns Review job to another Test Engineer?

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

What are Review Ethics?

1. Always review the content and not the author.
2. While reviewing spend time in identifying the mistakes and not the solutions for it.
3. Even after review, there are some mistakes means both author and reviewer are responsible for that.

Review Comments Report :

it is generally prepared by the reviewer once he / she identifies any mistakes present in author's test cases and finally the report is shared to the author

Fix Review Comments :

it is generally done by the author by referring the review comments report prepared by the reviewer and referring to this report only , author will fix / rectify all the mistakes done by him / her in the test cases

once the mistakes are rectifies , author will generate a new / modified copy of the test case and it is sent back to the reviewer

Verify The Fix :

it is generally done by the reviewer to check whether the previous mistakes present inside the test cases are properly fixed by the author or not and if the mistakes are all fixed , then reviewer will immediately inform to the test lead

Review Comments Report

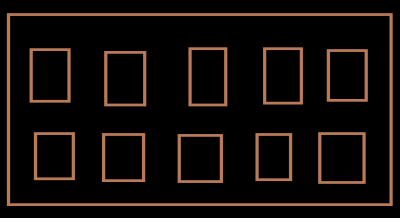
Sl No	Test Case Name	Reviewer		Author
		Comments	Severity	
1	SBI_AMOUNT_TRANSFER (**HEADER**)	Test case type attribute is missing	Major	Fixed
2	SBI_AMOUNT_TRANSFER (**BODY- STEP NO 22**)	The scenario has been repeated	Critical	Fixed
3	SBI_AMOUNT_TRANSFER (**FOOTER**)	Author's name is missing	Major	Fixed

Test Case Approval :

it is generally done by the test lead only when he / she comes to know that there are no mistakes present in the author's test cases (approval means test case is ready for execution)

Store Test Cases in Repository :

it is generally done by the indivisual test engineers only when their test cases are approved by the test lead and the repository is also called as MASTER TEST CASE REPOSITORY



Interview Tips

In interview, when the interviewer asks "how do you review a test case and what do you review in a test case?"

Always answer should start with – Body of test case, then header and finally template.

INTERVIEW QUESTIONS

1) ***What is the duration of your current project?***

   Ans) 8months – 1.5years. Whatever projects you put, be prepared to answer about any project. Always tell – "by the time i joined, 2major releases were over. I joined the project during the 3rd release and I have spent around 8months here".

2) ***Totally, in your current project, how many screens (features) are there?***

   Ans) an average complex application will have about 60 – 70 screens. A simple application like ActiTime has around 20 – 30 screens. So tell about 60-70 screens.

3) ***Totally, how many test engineers are there in your current project?***

   Ans) For 70 screens, 10 – 15screens / engineer. 70/15 = 5engineers. So, you can tell anywhere between 3 – 8 test engineers.

4) ***Totally in your current project, how many test cases are there in your current project***

   Ans) For 1 screen – you can write 10 – 15test cases (includes FT, IT, ST) 70 * 15 = 1050. You can tell anywhere from 800 – 1050 – 1200 test cases. Test case means 1 entire document with header, footer and many scenarios.

5) ***Totally, how many test cases you have written in your current project?***

Ans) This includes for all 3releases. You have joined in 3rd release. In 2releases, they would have written around 700test cases (worst case – 650test cases). You wrote in 3rd release test cases, so 550/5 = 110. You can tell anywhere between 80 -110 test cases (maximum of 240 also).

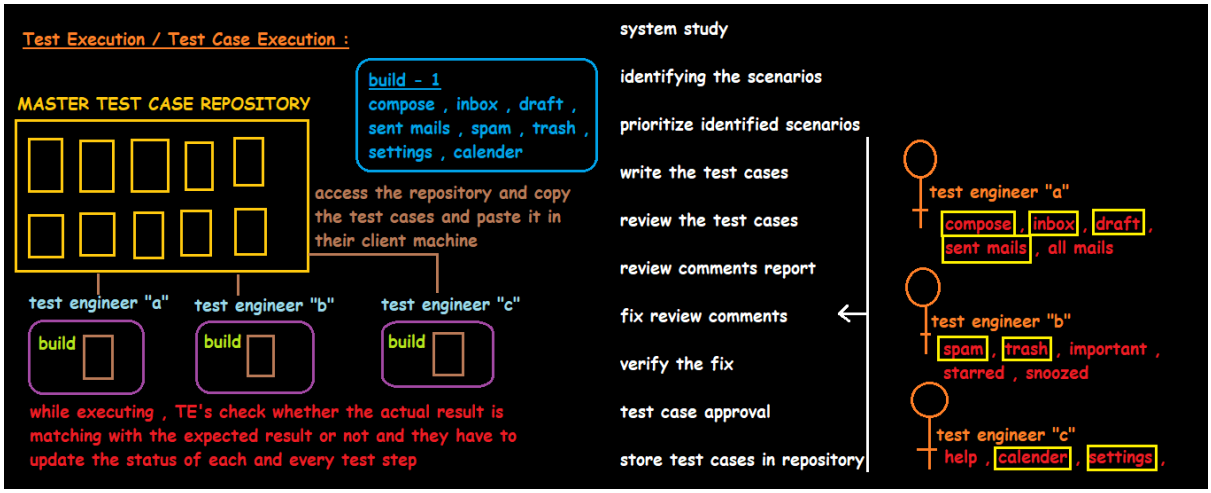
6) ***How many test cases can you write per day?***

Ans) you can tell anywhere between 3-5test cases. 1test case – 1st day, 2nd day 2test cases – 3rd day, 4th day 4test cases – 5th day 8 – 9test cases – 18th day Always answer like this, "initially, I used to write 3-5test cases. But, later stages, I started writing 7 – 8test cases because, □ Knowledge about the product became better □ I started re-using the test cases (copy and paste) □ Experience on the product Each test case that I write would generally have 20 -40 steps.

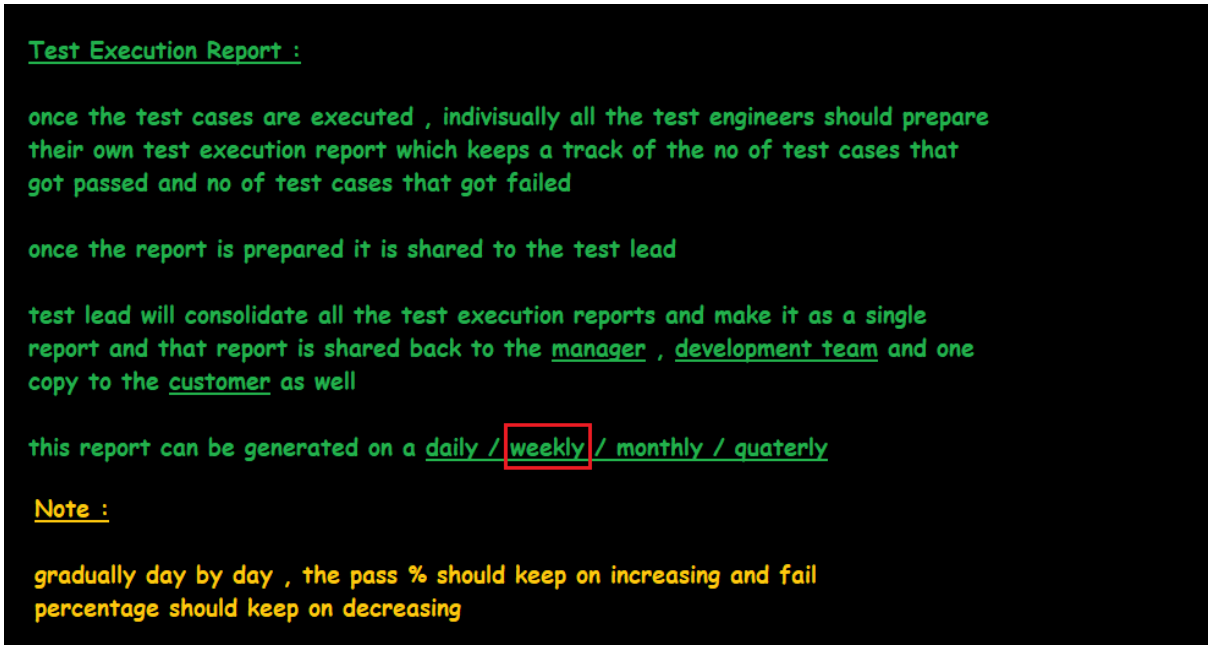
7) How you spend 10months in a project?

Ans) 1st 3days – no work. Next 2weeks – understand the product looking at the SRS. Next 2-3months – you wrote test cases for some features, review other's test cases and check and correct your reviewed test cases. By the end of 3months, developers give build no.1 – next 7months execute the test cases, find bugs – report it to developers – developers give new builds – you execute and find defects. Here, 7months time is spent. (Thus, around 10months you have spent).

Test Case Execution:



Test Execution Report:



Total No of Test Cases Present	Total No of Test Cases Executed	Total No of Test Cases Not Executed	Total No of Test Cases Passed	Total No of Test Cases Failed	% Pass	% Fail	
200	60	140	50	10			TEST ENGINEER "A"

Total No of Test Cases Present	Total No of Test Cases Executed	Total No of Test Cases Not Executed	Total No of Test Cases Passed	Total No of Test Cases Failed	% Pass	% Fail	
400	100	300	70	30			TEST ENGINEER "B"

Total No of Test Cases Present	Total No of Test Cases Executed	Total No of Test Cases Not Executed	Total No of Test Cases Passed	Total No of Test Cases Failed	% Pass	% Fail	
300	150	150	130	20			TEST ENGINEER "C"

Total No of Test Cases Present	Total No of Test Cases Executed	Total No of Test Cases Not Executed	Total No of Test Cases Passed	Total No of Test Cases Failed	% Pass	% Fail	
900	310	590	250	60			TEST LEAD

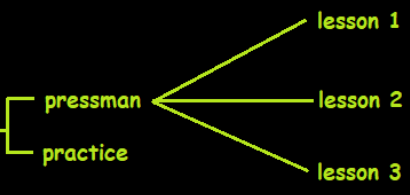
Test Case Design Techniques:

Test Case Design Techniques :

these test case design techniques are ued to identify the functionality scenarios (valid and invalid data)

Types of Test Case Design Techniques :

1. Error Guessing
2. Equivalence Partition
3. Boundary Value Analysis



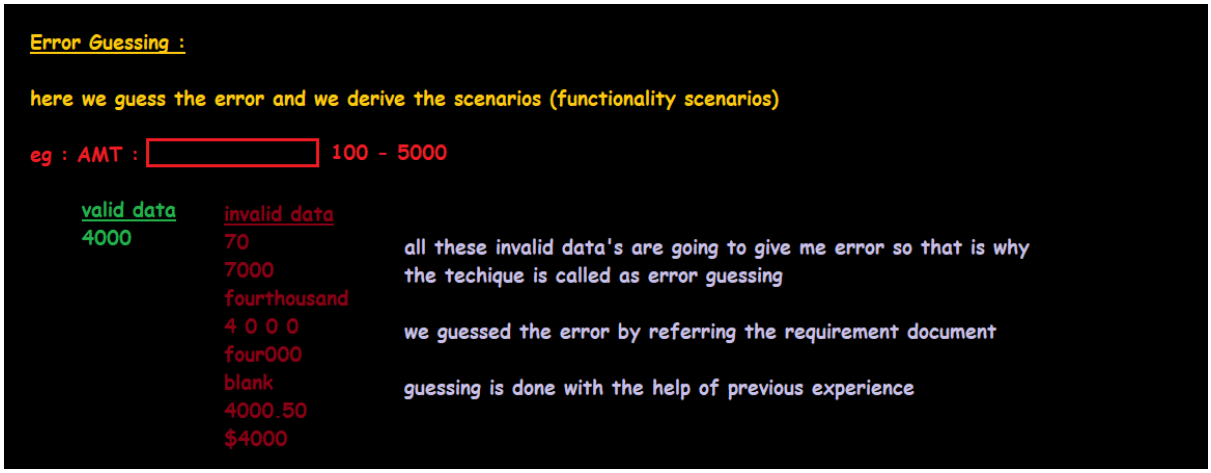
Types of Test Case Design Techniques:

1. **Error Guessing**
2. **Equivalence Partition**
3. **Boundary Value Analysis**

Error Guessing

Guess the error and derive the scenarios, we call it as error guessing.

We guess the error based on previous experience.



Equivalence Partition

It is of 2 types:

1. Pressman
2. Practice

Pressman

There are 3 lessons.

Equivalence Partition :
Pressman :

Lesson - 1

whenever input is given in the form of range of values then design the test cases for one valid and two invalid values

eg : AMT : [ ] 100 - 5000

valid data	invalid data
4000	70
	7000

whenever testing time is less we go for Lesson - 1
here chances of finding defects will be less because less scenarios are covered

Lesson - 2

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

eg : Keyboard ----> 100 units	50	0 , 120
Monitor ----> 80 units	50	0 , 90
Mouse ----> 60 units	40	0 , 80

real time example : any e-commerce website when the no of units left is less

Lesson - 3 :

whenever input is given in the form of boolean values , then design the test cases for both true value as well as false value

eg: ○Male ○Female

we have to check for both true condition as well as false condition i.e.
both cannot be true and both cannot be false

Male will be true and Female will be false
Female will be true and Male will be false

Lesson 1

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

Lesson 2

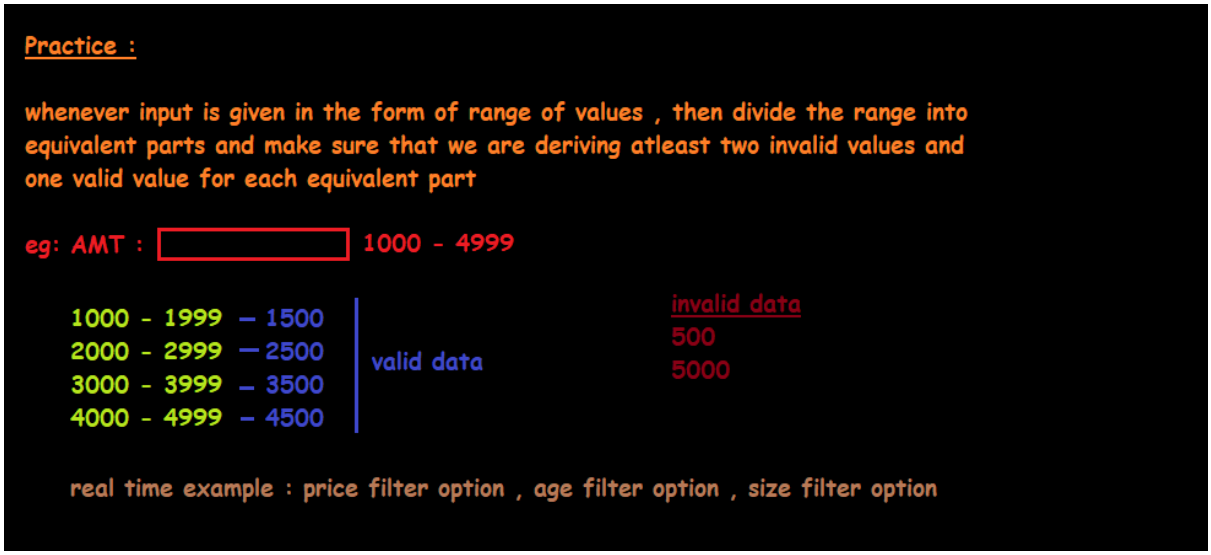
When the input is given in the form of set of values, then design the test cases for one valid and two invalid values.

Lesson 3

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

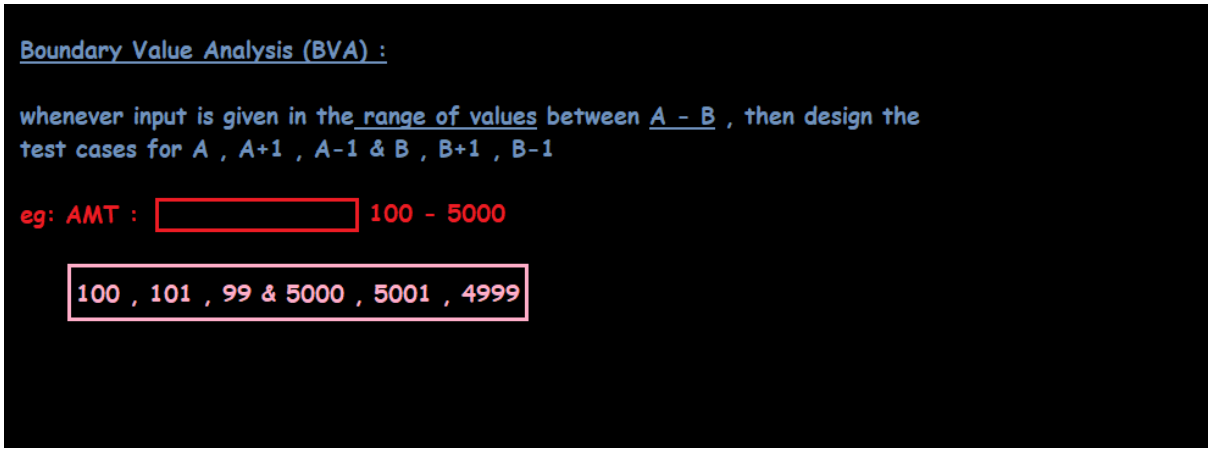
Practice

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.



Boundary Value Analysis

When input is given in the range between A-B, then design the test cases for A,A+1,A-1 and B,B+1,B-1.

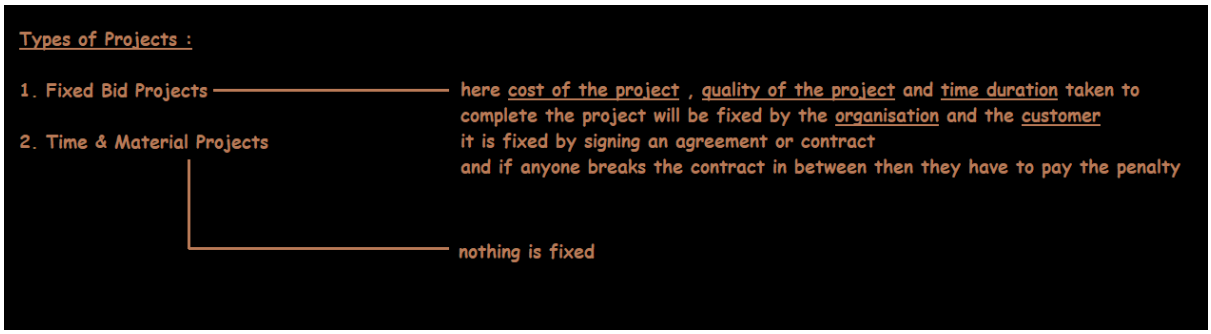


Types of Project

1. Fixed Bid Project
2. Time & Material Project

In fixed bid projects the duration and cost of the project is fixed by the customer and the software organization by signing an agreement which we call as **Service Level Agreement** (SLA). If anyone tries to break the agreement in between then penalty will be charged.

Whereas in time and material project nothing is fixed and no agreement is signed because there is a mutual understanding between the customer and software organisation.



What is the difference b/w defect, bug, error and failure?

Defect- If the feature functionality is not working a/c to the client given requirement.

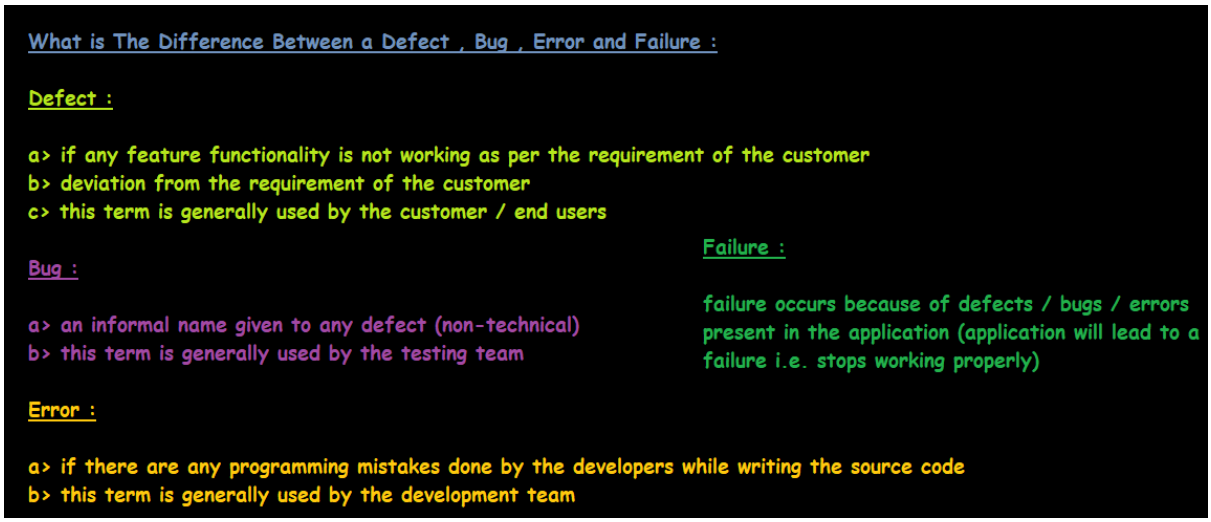
OR Deviation from the requirement.

OR The variation between the actual results and expected results is known as defect.

Bug- The informal name given to any defect is called Bug.

Error- it is a mistake done in the program because of which we are not able to compile or run the program

Failure- defect/bug/error in the application leads to failure or defect/bug/error causes failure.

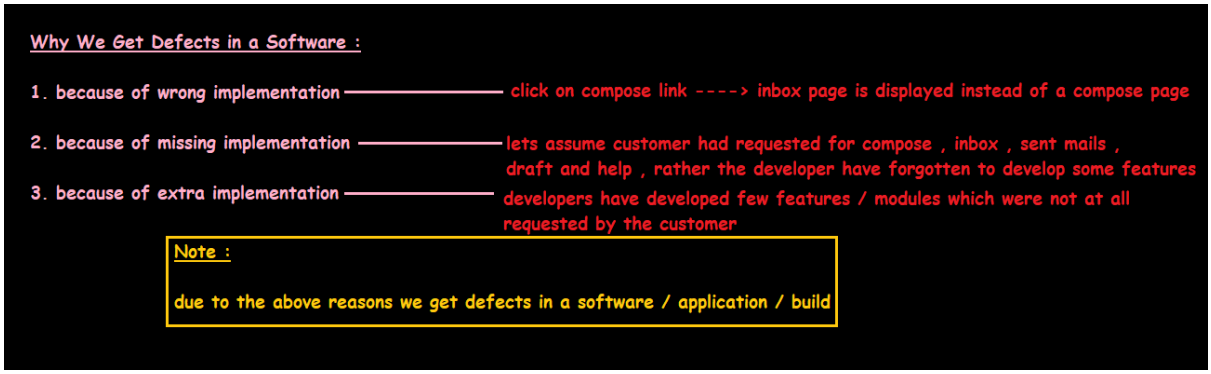


A bug occurs only because of the following reasons,

 Wrong implementation: - Here, wrong implementations means coding. For ex, 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.

 Missing implementation: - We may not have developed the code only for that feature. For ex, open the application – "SALES" link is not there only – that means feature has not been developed only – this is a bug.

 Extra implementation: That means something is not requested by the customer and it is developed by the developers.



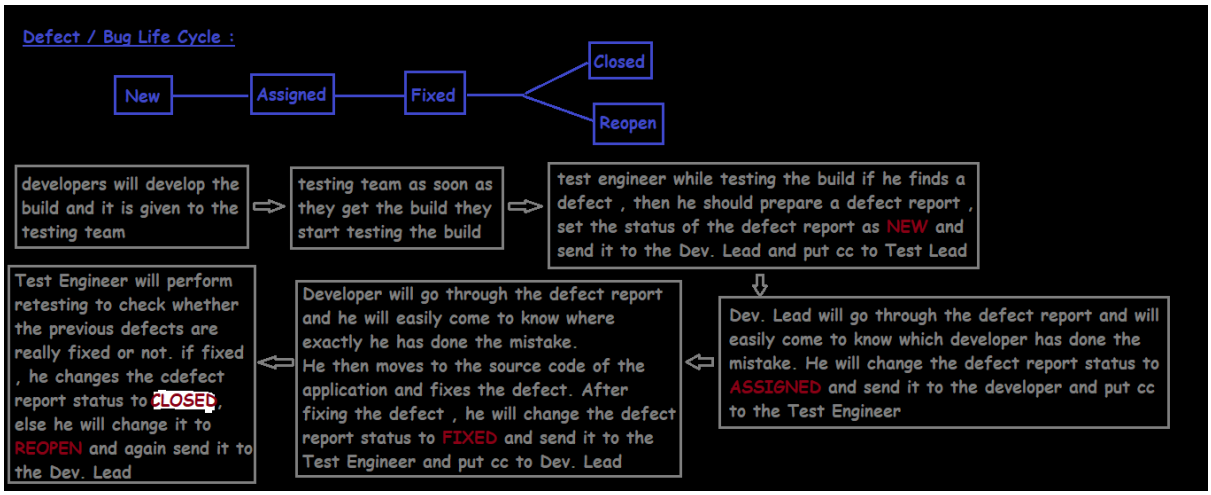
Defect Life Cycle / Bug Life Cycle

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 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 in 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 retesting 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 defect 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.



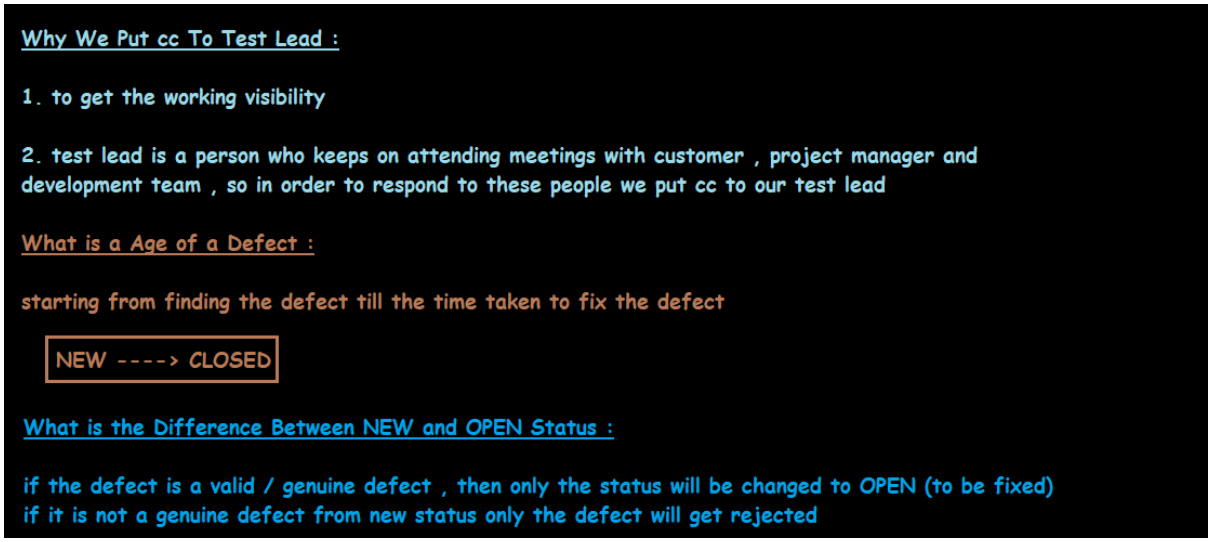
Why do Test Engineer put Cc to test Lead?

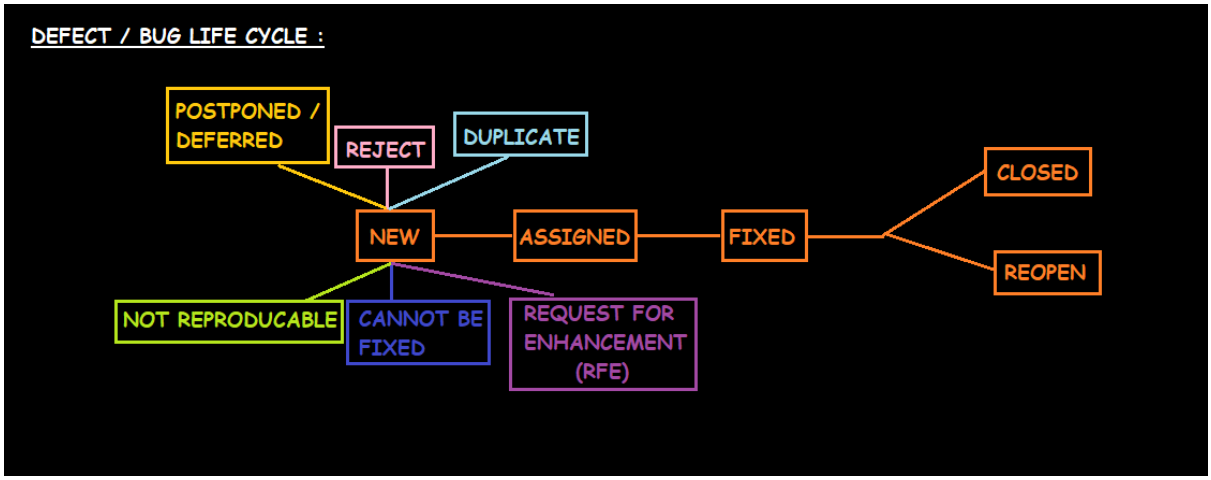
Because, test lead is the person who keeps on attending meetings with the development team and customer. So he should be aware of what exactly is happening in the project.

And to get the working visibility of the test engineers.

What is age of the defect?

Time duration taken from identifying the defect till it gets fixed.





"REJECT" STATUS

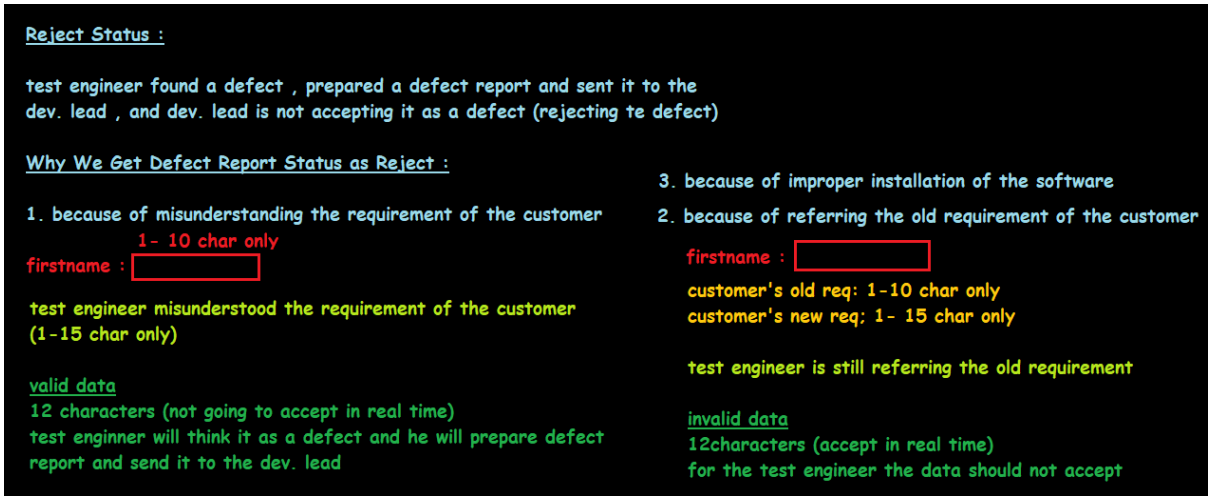
Now, when the TE sends a defect report – the Development Lead will look at it and reject the bug.

Bug is rejected because,

1) Misunderstanding of requirements

2) Referring to old requirements

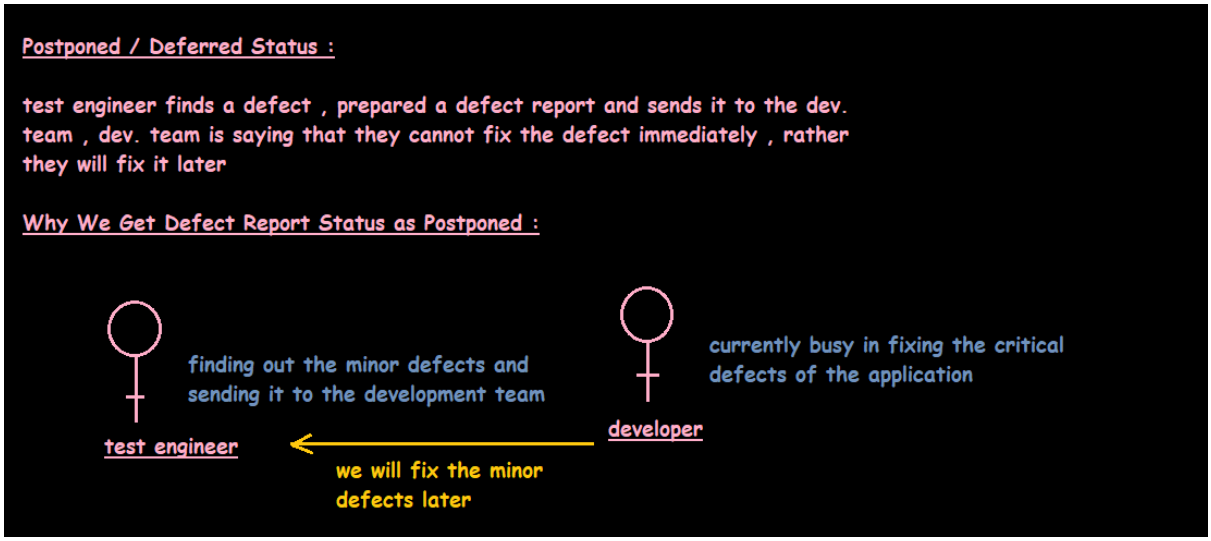
3) Wrong installation of build/product/software



POSTPONED STATUS

Whenever developers are fixing the critical defects, testing team is logging all the minor defects, in this scenario the development team will postpone the fixing of the minor defects.

We find a bug during the end of the release (could be major or minor but cannot be critical) – developers won't have time to fix the bug – such a bug will be postponed and will be fixed later or in the next release and the bug status will be "open"

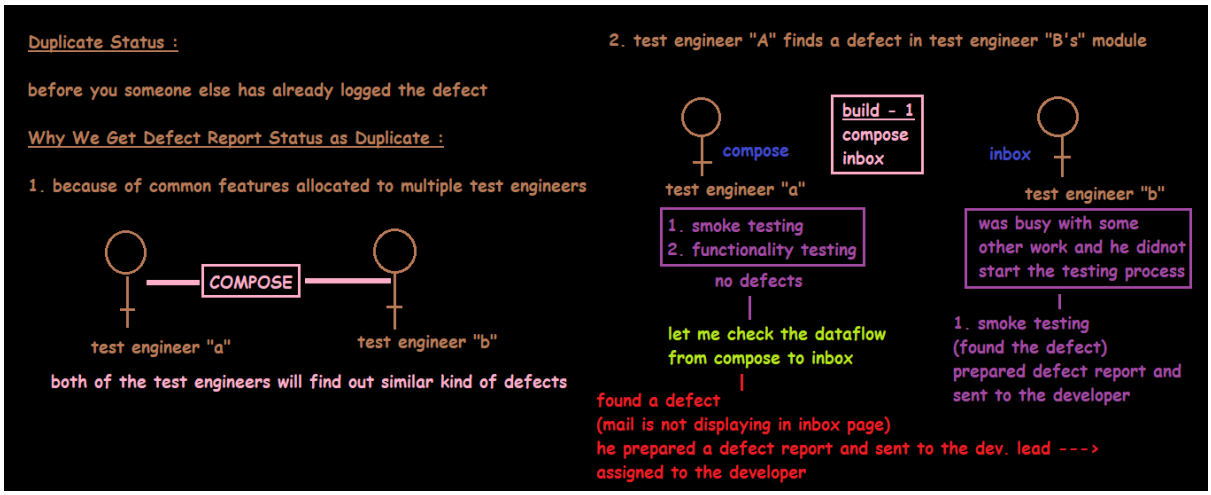


"DUPLICATE" STATUS

Before you someone else has already logged the defect.

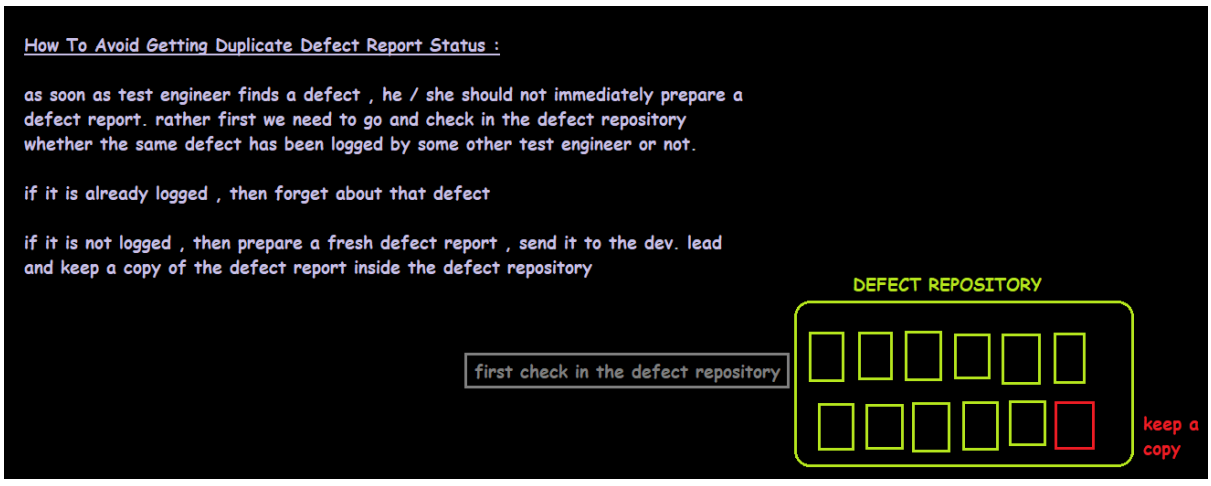
Why do we get duplicate bugs?

1. Because of common features
2. Test Engineer A finds a defect in Test Engineer B's module



How to avoid duplicate bugs?

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 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 T.E 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.

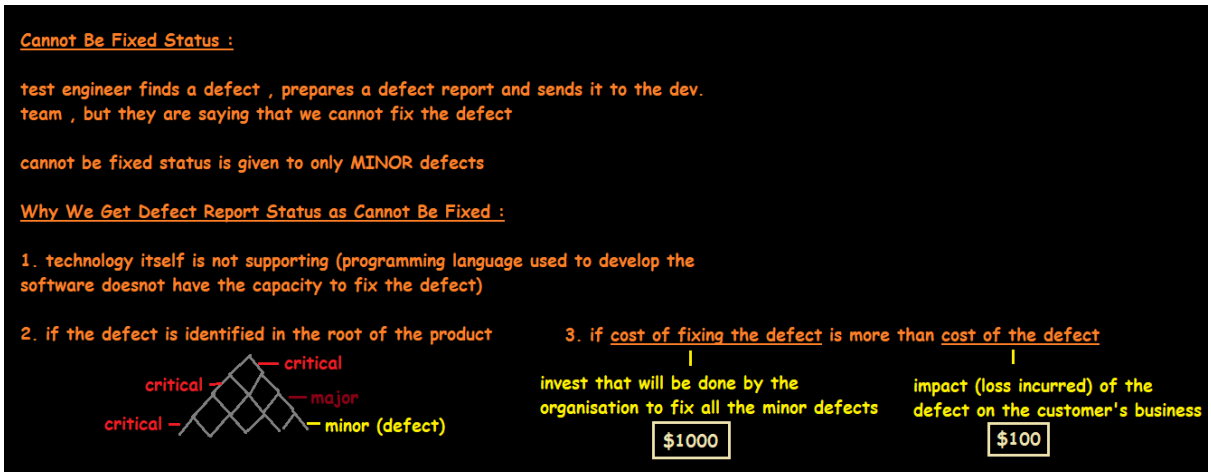


CANNOT BE FIXED

Chances are there – Test Engineer finds a bug and sends it to Development Lead – development lead looks at the bug and sends it back saying "cannot be fixed".

Why does this happen? – Because,

1. Technology itself is not supporting i.e. programming language we are using itself is not having capability to solve the problem
2. When a minor defect is identified in the root of the product. (If the developers are fixing that minor defect chances are there it might affect the entire application, so the developers don't take a risk of fixing that minor defect).
3. Cost of fixing the defect is more than cost of the defect.



NOT REPRODUCIBLE

Testing team finds a defect and send it to the development team, development team they are unable to reproduce/find that particular defect.

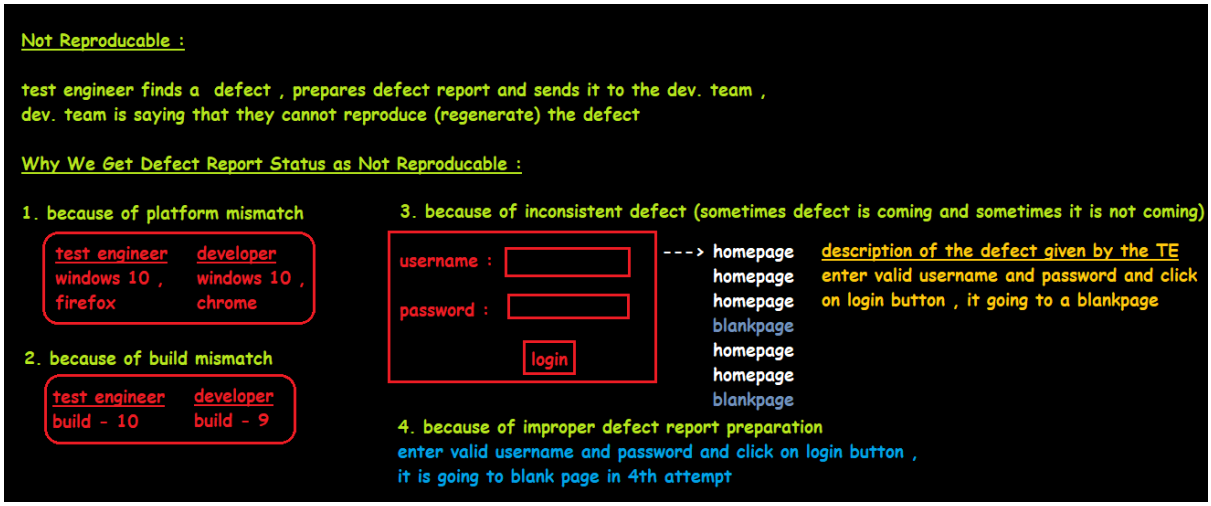
Why we get "not reproducible" defects?

1. Because of platform mismatch

2. Because of improper defect report

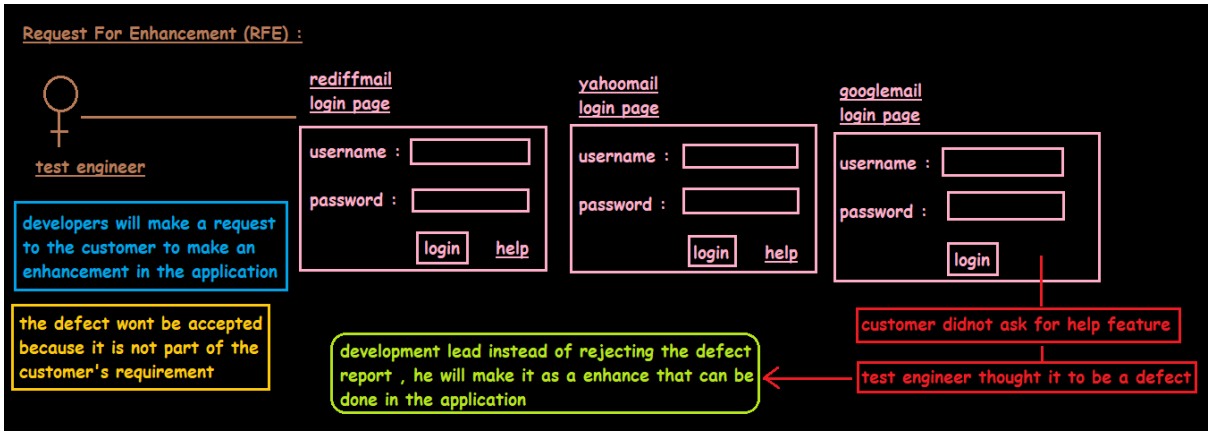
3. Because of build mismatch

4. Because of inconsistent defects



REQUEST FOR ENHANCEMENT (RFE)

Test engineer finds a bug and sends it to development team – when development team look at the report sent by the TE – They know it's a bug, but they say it's not a bug because it's not part of the requirement.



Defect ID	D_009
Module Name	Sent Mails
Test Case Name	Gmail_SentMails
Build No	B10
Test Environment	Windows 10 , Chrome
Status	New / Open / Assigned / Fixed / Reopen / Reject …
Severity	Blocker / Critical / Major / Minor
Priority	High / Medium / Low
Expected Result	Mails should be displayed in sent mails page
Actual Result	Mails is not displayed in sent mails page
Detailed Description	1. open the browser and enter the url 2. click on compose button 3. enter valid data in all the fields and click on send button 4. click on sent mails link
Found By	Test Engineer's Name

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

1. **Defect ID**
2. **Severity**
3. **Priority**

SEVERITY

Impact of the defect on the customers' business, we define as severity. To define severity we use the following terminologies:

1. Blocker / Showstopper
2. Critical
3. Major
4. Minor

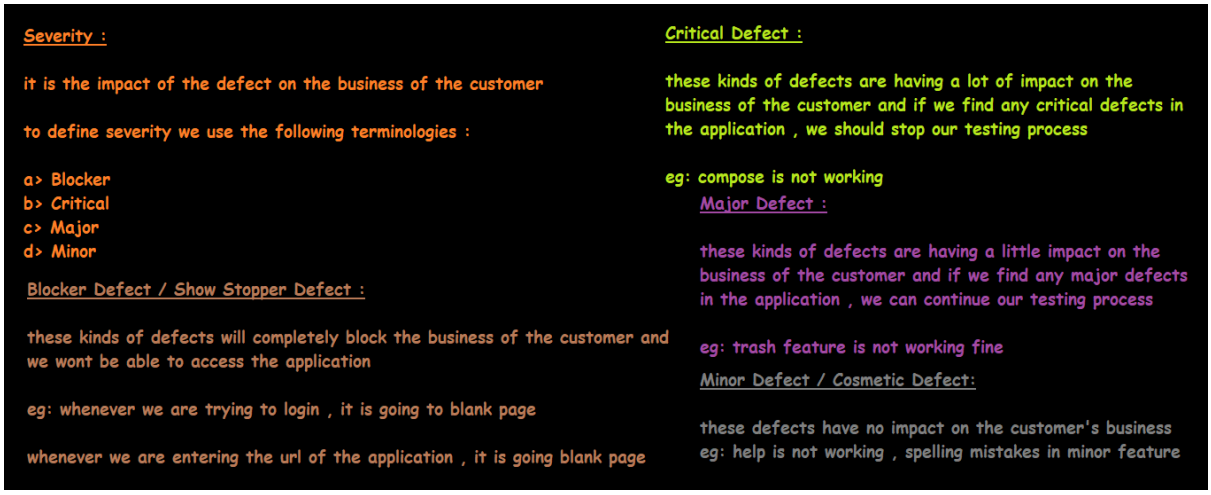
Blocker – The defect which is completely blocking the business of the customer.

Eg: Login or signup itself is not working in CitiBank application

Critical – A major issue where a large piece of functionality or major system component is completely broken. There is work around & testing cannot continue.

Major – A major issue where a large piece of functionality or major system component is not working properly. There is a work around, however & testing can continue.

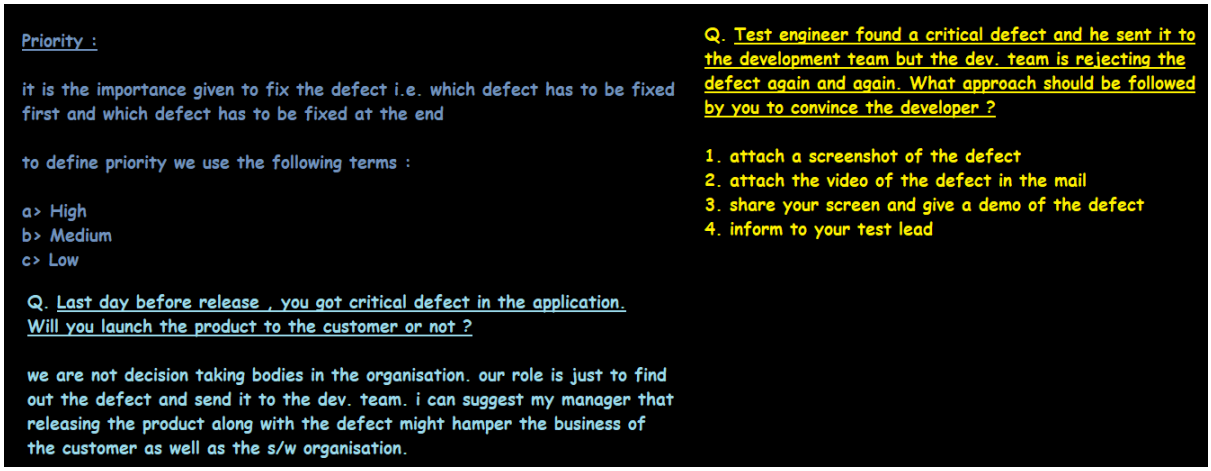
Minor – A minor issue that imposes some loss of functionality, but it is acceptable. For eg. Spelling mistakes in minor features.



PRIORITY of a Bug

It is the importance to fix the bug (OR) how soon the defect should be fixed (OR) which are the defects to be fixed first.

- High – This has a major impact on the customer. This must be fixed immediately.
- Medium – This has a major impact on the customer. The problem should be fixed before release of the current version in development
- Low – This has a minor impact on the customer. The flow should be fixed if there is time, but it can be postponed with the next release.



Who sets the Severity and Priority?

Test engineers generally sets the severity and priority, but the priority can be changed by the development team.

SOFTWARE TESTING LIFE CYCLE (STLC)

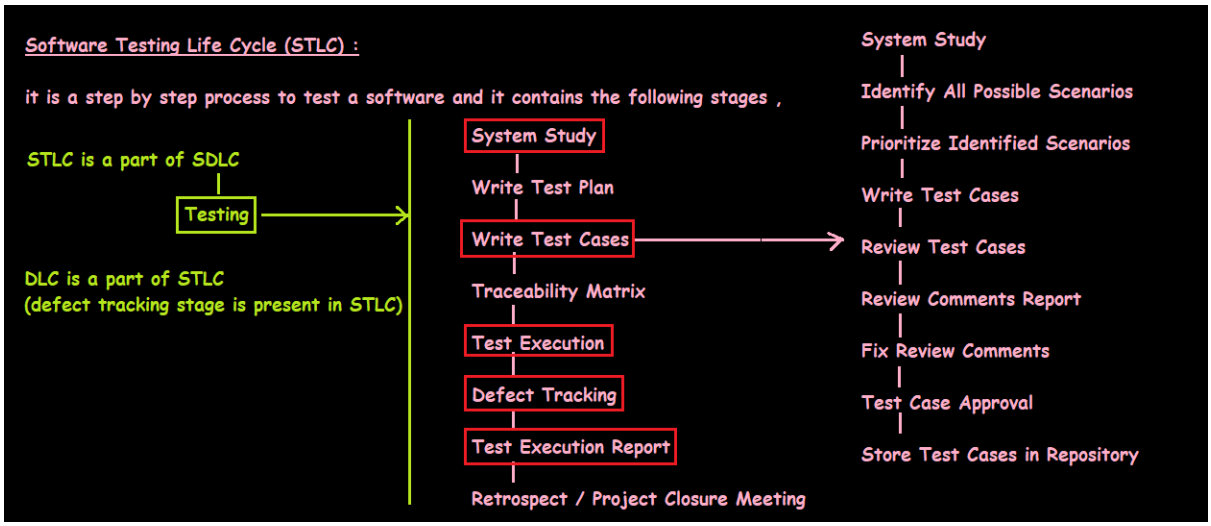
It is a procedure to test an application/software/build.

STLC is part of SDLC.

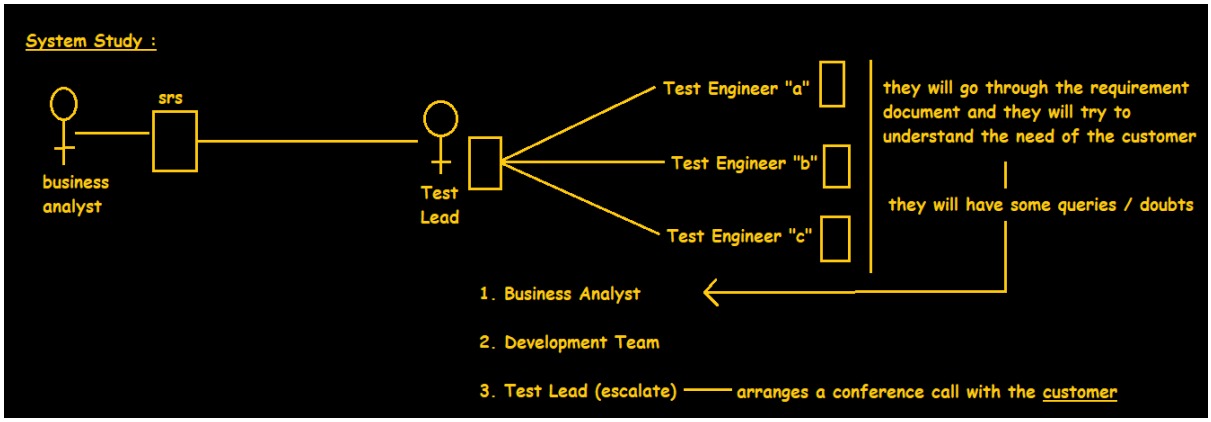
Defect Life Cycle is a part of STLC.

It contains several stages like:

1. **System Study**
2. **Write Test Plan**
3. **Write Test Cases**
4. **Traceability Metrics**
5. **Test Execution**
6. **Defect Tracking**
7. **Test Execution Report**
8. **Retrospect Meeting**



System Study:

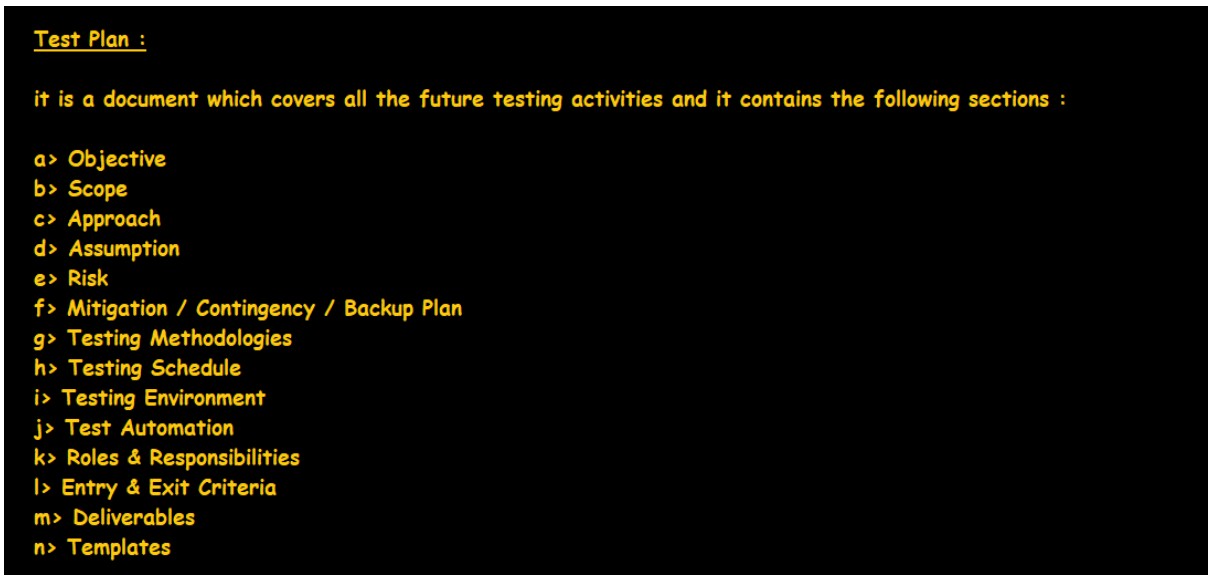


Test Plan:

Test plan is a document which drives all future testing activities.

Test plan is prepared by Test manager or by Test Lead

There are 14 sections in a test plan. We will look at each one of them below,



1) **OBJECTIVE**: - It gives the aim of preparing test plan i.e, why are we preparing this test plan.

2) **SCOPE**:-

2.1 Features to be tested

For ex, Compose mail, Inbox, Sent Items, Drafts

1.2 Features not to be tested

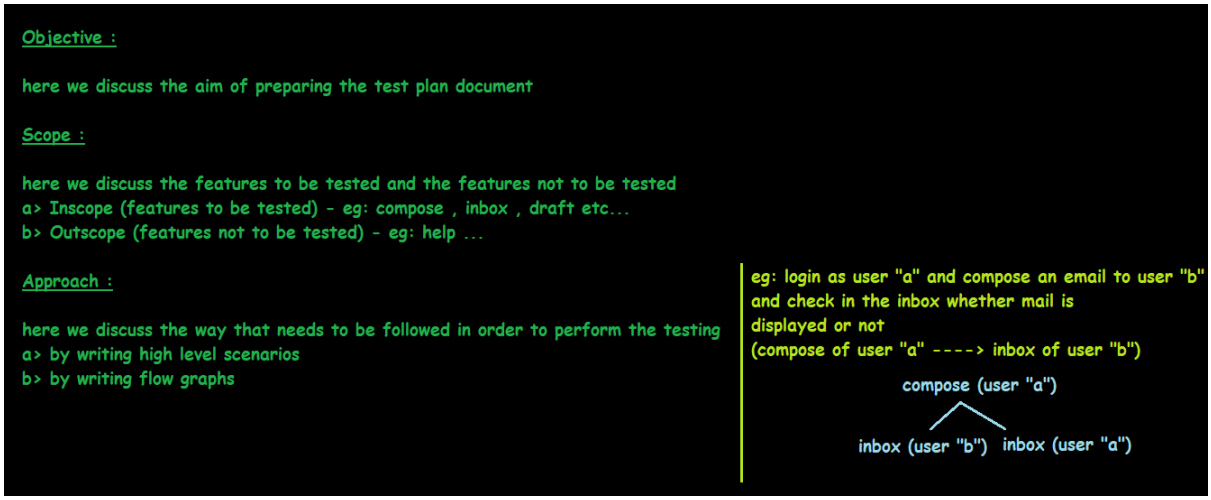
For ex, Help … … … …

3) APPROACH

The way we go about testing the product in future,

a) By writing scenarios

b) By writing flow graphs



4) ASSUMPTIONS

When writing test plans, certain assumptions would be made like technology, resources etc.

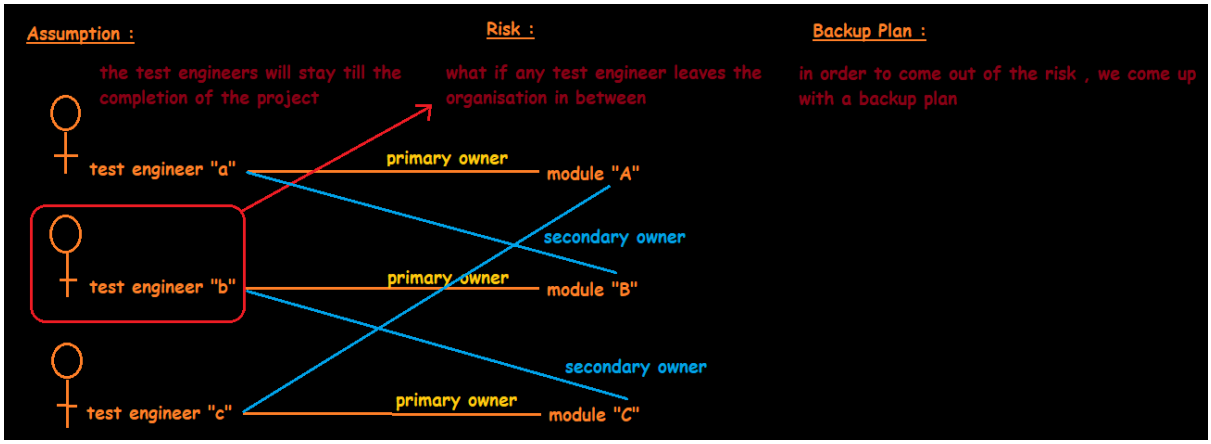
5) RISKS

If the assumptions fail, risks are involved

6) CONTINGENCY PLAN OR MITIGATION PLAN OR BACK-UP PLAN

To overcome the risks, a contingency plan has to be made.

In the project, the assumption we have made is that all the 3 test engineers will be there till the completion of the project and each are assigned modules A, B, C respectively. The risk is one of the engineers may leave the project mid-way. Thus, the mitigation plan would be to allocate a primary and secondary owner to each feature. Thus, one engineer quits - the secondary owner takes over that particular feature and helps the new engineer to understand their respective modules. Always assumptions, risks, mitigation plan are specific to the project.



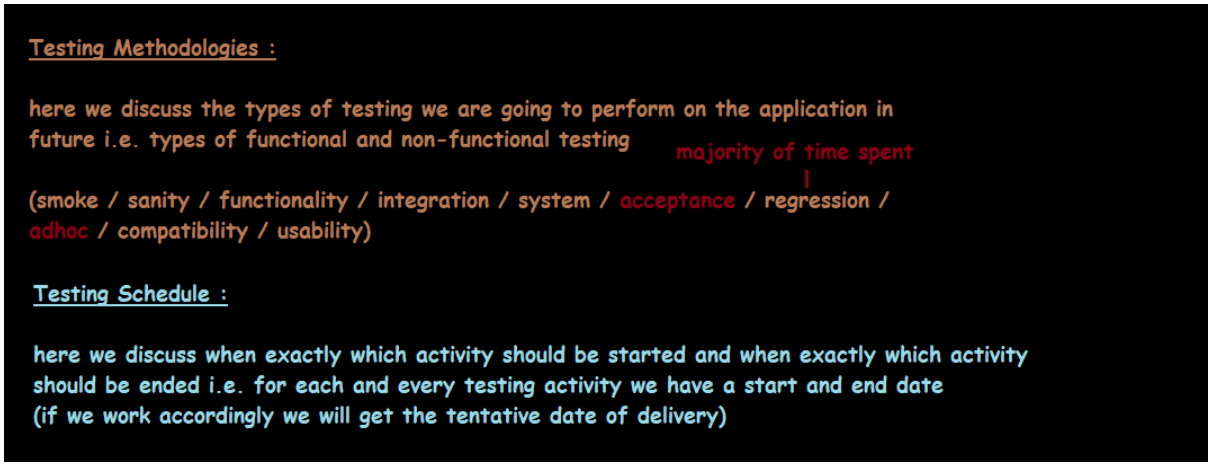
7) TESTING METHODOLOGIES (Types of Testing):

Depending upon the application, we decide what type of testing we do for the various features of the application. We should also define and describe each type of testing we mention in the testing methodologies so that everybody (dev team, management, testing team) can understand, because testing terminologies are not universal.

For example, we have to test www.flipkart.com, we do the following types of testing, Smoke testing Functionality testing Integration testing System testing Adhoc testing Compatibility testing Regression testing Usability testing.

8) TEST SCHEDULES:-

This section contains – when exactly which activity should start and end? Exact date should be mentioned and for every activity, date will be specified.



9) TEST ENVIRONMENT:

Here we discuss the environment / platform (hardware & software) which should be used in order to test the application / build in future.

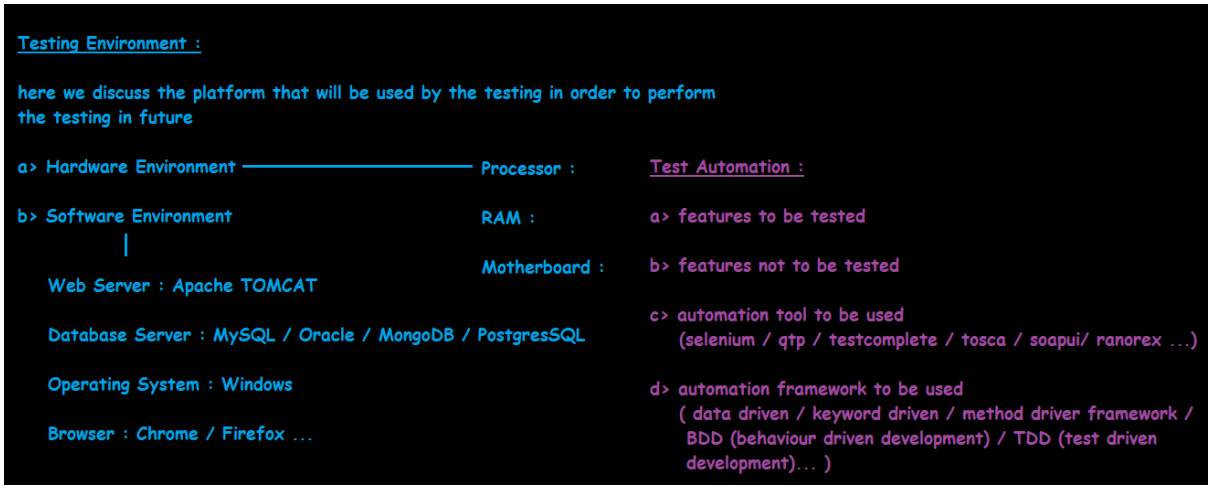
10) TEST AUTOMATION:

10.1 Features to be automated

10.2 Features not to be automated

10.3 Which is the automation tool we are planning to use

10.4 What is the automation framework we are planning to use?



11) ROLES AND RESPONSIBILITIES

11.1 Test Manager

1. Writes or reviews test plan

2. Interacts with customer, development team and management

3. Handle issues and escalations

4. Sign the release Note

11.2 Test Lead

1. Writes or reviews test plan

2. Interacts with development team and customers

3. Allocates work to test engineers and ensure that they are completing the work within the schedule

4. Consolidate reports sent by Test Engineers and communicate it to development team, customers

11.3 Test Engineer

1. Review test plan

2. Write, Review and Execute test cases

3. Write traceability matrix

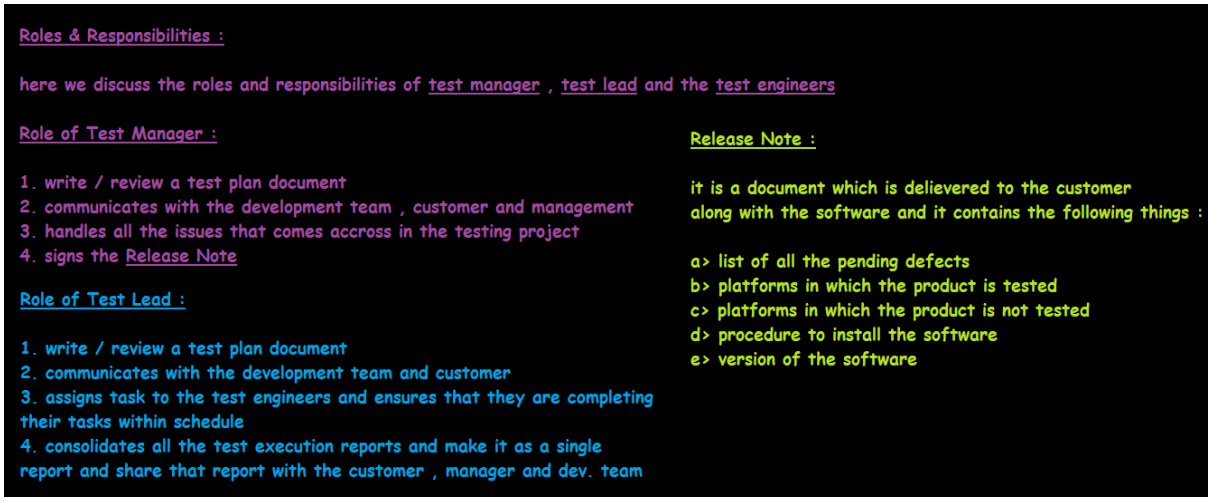
4. Perform different types of testing on the application

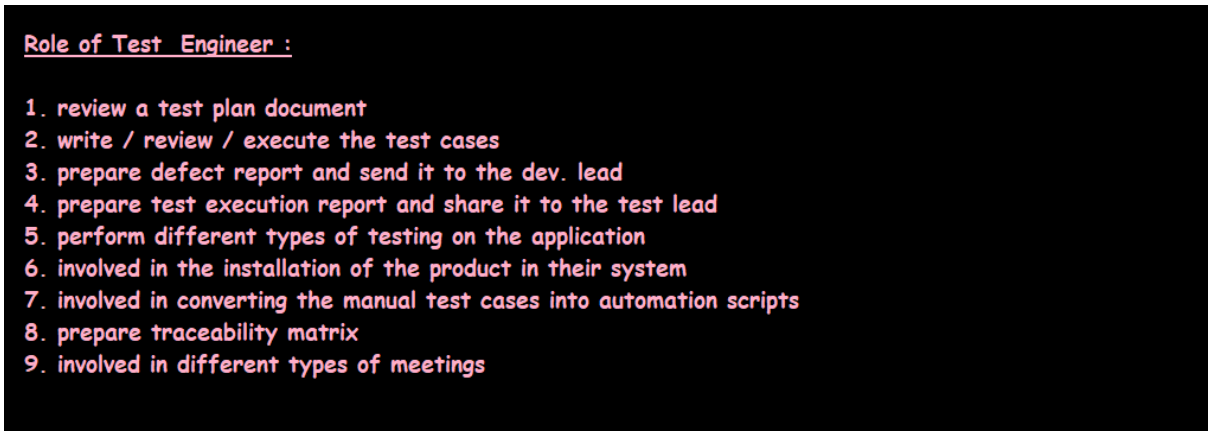
5. Prepare test execution report and communicate it to Test lead.

6. Prepare defect report and send it to development team.

7. Convert manual test cases into automation scripts.

8. Involved in installation and setup of software.





12) ENTRY AND EXIT CRITERIA:

Entry criteria for FT :

a) WBT should be over

b) Test cases should be ready

c) Smoke testing should be done

d) Resources should be available

Before we start with Functional Testing, all the above entry criteria should be met.

After we are done with FT, before we start with Integration Testing, then the exit criteria of FT should be met.

The testing team would have decided that in order to move onto the next stage, the following criteria should be met,

Exit criteria for FT :

1. There should not be more than 20 critical bugs

2. There should not be more than 50 major bugs

3. There should not be more than 100 minor bugs.

If all the above are met, then they move onto the next testing stage.

Entry criteria for IT :

a) should have met exit criteria of FT
b) Test cases should be ready
c) Resources should be available

Exit criteria for IT :

a) %age pass for FT should be 90%, and the %age pass for IT should be 85%
(Only if the above condition is satisfied then only we can move out of integration testing and start another kind of testing)

Entry criteria for ST :

- Exit criteria of IT should be met

- Minimum bunch of features must be developed

- Test environment should be similar to production environment

- Test cases should be ready

- Resources should be ready

Exit criteria for ST:

- pass % should be 99%

- There should be 0 critical bugs.

There could be some 20minor bugs.

If all this is met, then product can be released.

Note: All the numbers given above are just for example sake. They are not international standard numbers!

Entry & Exit Criteria :

here we discuss what conditions needs to be satisfied in order to start a kind of testing and
what conditions needs to be satisfied to stop one kind of testing and start another kind of testing

Entry Criteria to do Functionality Testing :

1. White Box Testing should be over
2. Smoke Testing should be over
3. Test cases should be ready
4. Resources (test engineers) should be available

Exit Criteria to do Functionality Testing :

1. there should not be more than 20 critical defects
2. there should not be more than 50 major defects
3. there should not be more than 100 minor defects

Entry Criteria to do Integration Testing :

1. should have met the exit criteria of Functionality Testing
2. Test cases should be ready
3. Resources should be available

Exit Criteria to do Integration Testing :

1. functionality testing pass % should be 90 %
2. integration testing pass % should be 85 %

| test execution report

Entry Criteria to do System Testing :

1. should have met the exit criteria of Integration Testing
2. minimum bunch of features should be ready
3. testing environment should be similar with production environment
4. test cases should be ready
5. resources should be available

Exit Criteria to do System Testing :

1. there should not be any critical defects (there can be major and minor defects)
2. software pass % should be 99%

13) DELIVERABLES

It is the output from the testing team. It contains what we will deliver to the customer at the end of the project.

It has the following sections,

13.1 Test Plan

13.2 Test Cases

13.3 Test Scripts

13.4 Release Note

13.5 Defect Report

13.6 Test Execution Report

14) TEMPLATES

This section contains all the templates for the documents which will be used in the project. Only these templates will be used by all the test engineers in the project so as to provide uniformity to the entire project. The various documents which will be covered in the Template section are,

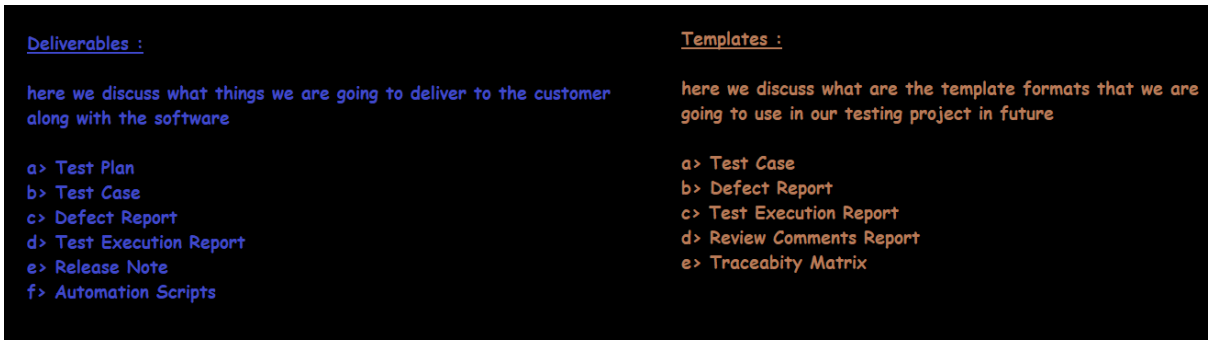
14.1 Test Case

14.2 Traceability Matrix

14.3 Test Execution Report

14.4 Defect Report

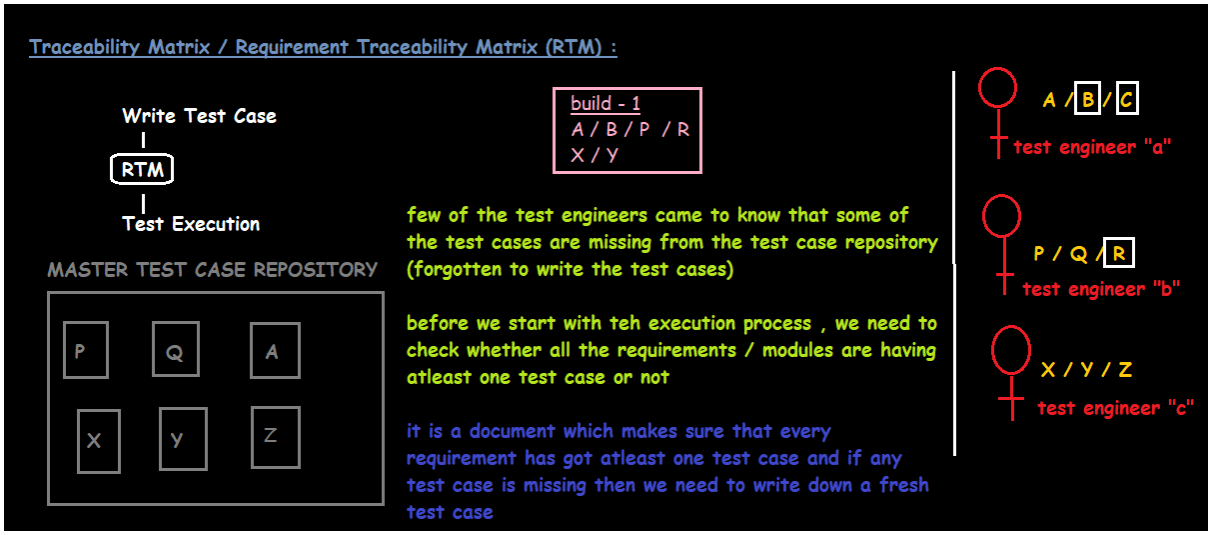
14.5 Review Comments Report



Write test case – we write test cases for each features (functionality, integration and system). These test cases are reviewed, and after all mistakes are corrected and once the test cases are approved – then they are stored in the test case repository.

(Here you can explain the procedure to write test cases)-present in the notes of test cases

Traceability Matrix – it is a document which ensures that every requirement has a test case. Test cases are written by looking at the requirements and test cases are executed by looking at the test cases. If any requirement is missed i.e., test cases are not written for a particular requirement, then that particular feature is not tested which may have some bugs. Just to ensure that all the requirements are converted, traceability matrix is written.



Defect Tracking – any bug found by the testing team is sent to the development team. This bug has to be checked by the testing team if it has been fixed by the developers.

Test Execution Report: - Send it to customer – contains a list of bugs (major, minor and critical), summary of test pass, fail etc. and when this is sent, according to the customer –

Either on daily weekly monthly quarterly half-yearly basis.

Retrospect meeting – (also called **Post Mortem Meeting / Project Closure Meeting**) The Test Manager calls everyone in the testing team for a meeting and asks them for a list of mistakes and achievements in the project.

So, whenever a new project comes, we will try to follow the achievements and avoid the mistakes done in the previous project.