

MANUAL TESTING FULL NOTES

SR.NO	TOPICS	PAGE NUMBER
1	SDLC	2 – 15
2	SOFTWARE TESTING	16 – 19
3	BLACK-BOX TESTING	20 – 40
4	NON – FUNCTIONAL TESTING	41 – 42
5	TYPES OF TESTING	43 – 45
6	1.TEST CASE 2.TEST CASE TEMPLATE 3. TEST CASE DESIGN TECHNIQUES 4.TEST CASE REVIEW PROCESS 5.PROCEDURE TO WRITE TEST CASE	46 – 47 48 – 49 50 – 51 52 – 53 54 – 55
7	SEVERITY & PRIORITY	56 – 58
8	DEFECT DEFECT/ BUG LIFE CYCLE & STATUS INTERVIEW QUE ON DEFECT	59 – 60 61 – 63 64
9	STLC	65 – 59

SDLC

- SDLC stands for Software Development Life Cycle.
- It is a step by step procedure or standard procedure to develop a new software.

There are 7 stages :

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

1) Requirement Collection:

- It is done by Business Analyst or Product

Analyst. There are two types of company:

- 1)Service based company and 2)Product based company

- Business Analyst will be present in service based company.

-Business Analyst from software company will go the customer place and collect requirements in customer requirement specification which will be in customer business language and come back to the software company and he will convert CRS into SRS.

- Product Analyst will be present in product based company.

-Product Analyst will do a market research, and prepare his own CRS and convert it into SRS.

2) Feasibility Study/ Analysis :

- It is done by the Business Analyst, Architect, Finance Team, HR Team and Product Manager. Business Analyst : He will convert CRS into SRS.

Architect : He will think Technical point of view, He will analyze whether Technology is available or not, to develop the software.

HR Team : HR Team will think from Hiring point of view, They will analyze whether Engineers are available or not to develop the software.

Finance Team : They will think from budget/ Money point of view, They will analyze whether money is available or not to develop the software.

Project Manager : He is like a Decision Maker.

3) Design :

There are two types of Design :

1)Level Design/ External Design and 2)Low Level Design/ Internal Design

1)High Level Design/ External Design:

-Designing the architecture of the software is called High Level Design.

-It is done by the Architect.

2)Low Level Design/ Internal Design:

-Designing the Every Feature in Detail is called Low Level Testing.

-It is done by the Architect/ SR. Developer.

4) Coding :

-Here developers will write a code/ Program in-order to develop the Software.

-It is done by :

SR. Developer – Critical

Features JR. Developer –

Major Features

Fresher Developer – Minor Features

5) Testing :

-Here Test Engineers will test the software in- order to find the defects in the software.

-It is done by :

SR. Test Engineer – Critical

Features JR. Test Engineer –

Major Features Fresher

Engineer – Minor Features

6) Deployment/ Installation :

-It is done by the Installation Engineer.

-From Software Company Installation Engineer will go to the customer place and install the software in Production Server.

7) Maintenance :

-Here Customer will sign the agreement with the company called as service legal agreement.

-If something happens with the software within the maintenance period, the company will fix it free- of-cost.

-After the Maintenance period is completed, If something happens to the software, the company will charge for it.

When to go for SDLC :

- Whenever Person or Company wants to develop a new software, The Should follow SDLC.

Models of SDLC :

1) Waterfall Model 2) Spiral Model 3) V Model 4) Prototype Model 5) Agile Model

1) Waterfall Model/ Traditional Model/ Sequential Model :

- It is a step by step procedure or standard procedure to develop the new software.
- There are 7 stages : Requirement collection, Feasibility Study/ Analysis, Design, Coding, Testing, Installation/ Deployment, Maintenance.
- NOTE: In early days there was no Test Engineer, Developers were only involved in Testing the Software.
- Currently the companies which follow Waterfall Model has a Test Engineer.

Drawbacks of Developers involved in Testing the Software?

- Developers will see the software from positive point of view, they never see the software from Negative point of view.
- Developers will have over confidence, that whatever he has developed, it will work.
- He will utilize all the Testing time in developing the Software.
- If there is a bug in a software developers might try to hide it.

Advantages of Waterfall Model:

- 1) Requirement and Design doesn't change, so we get stable product..
- 2) Software quality will be good.

Disadvantages of Waterfall Model :

- 1) Backtracking is not possible (e.g- If customer wants any changes in the requirement, it is not possible because in waterfall model after requirement collection is done the requirements are frozen)
- 2) Requirement and design is not tested.
- 3) Developers were involved in testing the software.

4) Requirement changes are not allowed, so the model is not flexible.

Application/ When to go for Waterfall Model :

-For simple and small project.

-For short time project.

-When customer doesn't ask for changes in the requirements, we go for waterfall model.

2)Spiral Model :

- It is step by step procedure or standard procedure to develop the software.
- Whenever there is dependency between the modules, we go for Spiral Model.
- In spiral Model customer will give the requirements in stages.

Advantages of Spiral Model :

- 1)Customer gets an opportunity to see the software after every cycle.
- 2)Customer get an opportunity to ask any changes in requirements after every cycle.
- 3)Testing happens in every cycle before going to next cycle.
- 4)Software quality will be good.

Disadvantages of Spiral Model:

- 1)Requirement changes is now allowed, in between the cycle. 2)Requirement and design is not tested.
- 3)Same process keeps on Repeating.
- 4)There also developers involved in testing the software.

Application/ When to go for Spiral Model :

-Whenever there is dependency between the modules, we go for spiral model.

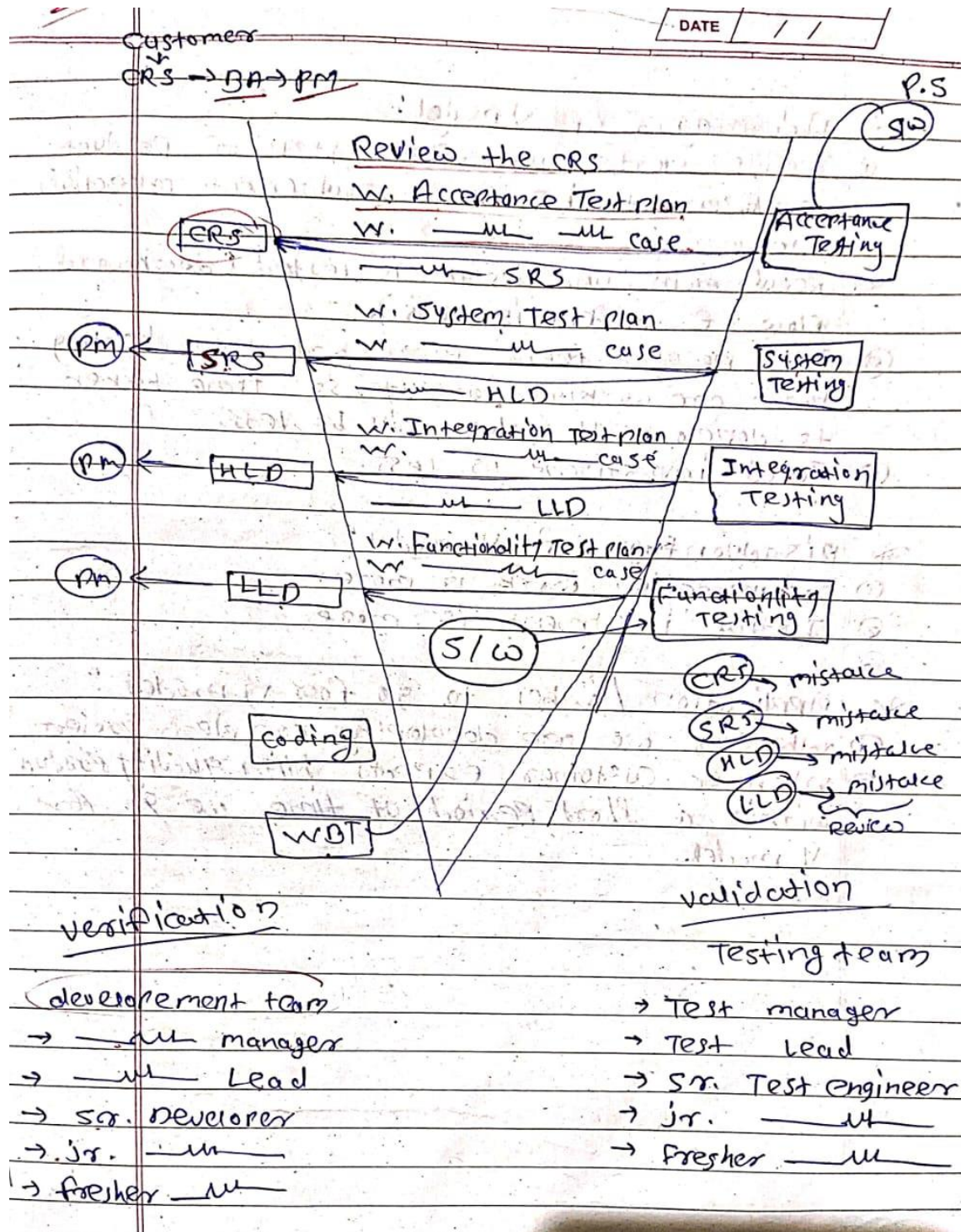
-Whenever customer gives the requirements in stages, we go for spiral

model. Requirement changes can be handled in two ways :

- 1) Major Changes – If customer asks 70% - 90% changes in requirement.
- 2) Minor Changes – If customer asks 15% - 20% changes in requirement.

3) V Model/ V&V Model/ Seizer Model/ Fish Model :

NOTE : To overcome the drawbacks of the waterfall model and spiral model, i.e. in waterfall model Requirement changes are not allowed and in both, waterfall model and spiral model Requirement and design is not tested.



Verification :

The process of reviewing Requirements(CRS,SRS), Design(HLD,LLD), Coding and all other related documents is called as verification.

Validation :

It is actual testing done once after software is developed, Here Test Engineer will test the software in order to find the defects in the software.

Advantages of V Model:

- Testing starts from the early stage of product development. i.e. From requirement collection stage.
- Requirement and Design is tested.
- Both Development team and Testing team are working parallelly, So time taken to develop the software will be less.
- Total investment is less.

Disadvantages of V Model :

- Documentation work is more.
- Initial investment is more.

Application/ When to go for V Model :

- Whenever we are developing complex project.
- Whenever customer expects high quality product within a short period of time, we go for V Model.

4) Prototype Model/ Dummy Model :

-Prototype model is a dummy model prepared by a web developers or content developers, where in they convert text format into image format, by using tools like paint, adobe Photoshop , HTML,CSS.

Stages of prototype model :

- 1.Requirement collection
- 2.Feasibility study/ Analysis
- 3.Design and Development of prototype
- 4.Prototype Testing
- 5.Customer Review
- 6.Design
- 7.Coding
- 8.Testing
- 9.Installation
- 10.Maintanance

Prototype Testing : Testing done to check whether Text format is converted into image format or not.

Actual Testing : It is testing, done once after software is developed. Here, test engineer will test the software in order to find defect in the software.

Advantages of Prototype Model :

- Customer gets an opportunity to ask for changes in the early stage.
- Customer get an opportunity to see the software in early stage.

Disadvantages of Prototype Model :

- Time taken will be more.
- There investment is needed just a build a prototype.
- There will be delay it's starting actual product development.

Applications/ When to go for Prototype Model :

- If customer is new to the business.
- When customer don't know, How to give the requirement we go for Prototype Model.

5. Agile Model :

- Agile is a model wherein we develop the software in an incremental and iterative process.
- They came up with this model in order to overcome the drawbacks that were there in the traditional model. Here we build large products in shorter cycles called Sprint.

Scrum:

It is the process used to build an application in agile model.

Scrum team:

It is the group of Engineers working towards completing committed features or stories.

- a. Generally, scrum team will have 7-12 members.
- b. It includes Shared team and Core team members
- c. Core team includes Scrum master, Development Engineer and Test Engineer.
- d. Shared team includes Architect, Product Owner, Database admin, network admin, UI and UX designers, BA.
- e. Scrum master leads this entire scrum team and he facilitates everyone to complete their task.

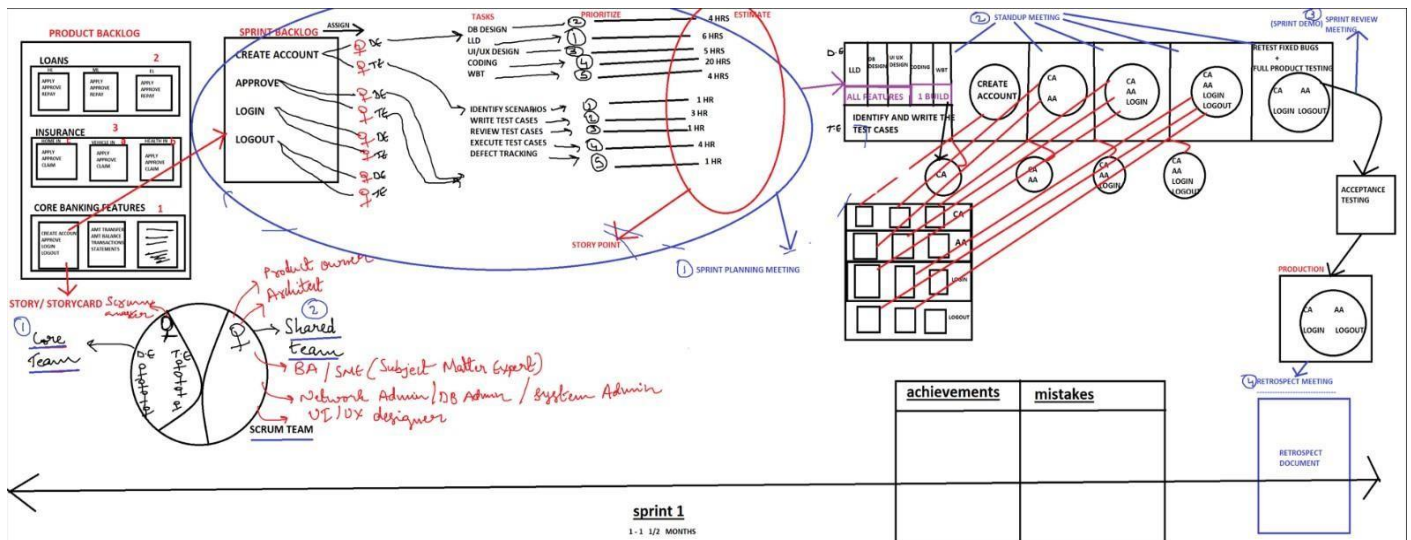
Product Backlog:

It is a prioritized list of stories or requirements that must be developed in the complete project.

- a. Generally, Product owner, Customer, Business analyst, architect, Scrum master will be involved in building it.
- b. Generally, stories in product backlog need not be in detail.

Sprint Backlog:

It is a list of stories and the associated task committed by the scrum team that must be delivered within one sprint.



I. Sprint Planning Meeting:

- Here the entire scrum team sits together and pulls the stories from the product backlog.
- Scrum master assigns each story to development engineer and test engineer.
- Now each engineer derives the tasks to be completed to build the stories.
- Each engineer will estimate the time taken to complete each task i.e. they derive the story point.

Following are the roles played by different people in Sprint planning meeting:

1. Scrum master:

- This complete meeting is driven by the scrum master.
- His prime role is to facilitate the complete meeting and co-ordinate between all stakeholders

2. Product owner:

- He clarifies if there are any questions related to stories or requirements.

3. Development Engineer:

- He should derive the task for building every story.
- He prioritizes which story to be built first and which story to build later in the sprint.
- He prioritizes the tasks.

d. He derives the story point.

4. Test engineer:

a. He derives the task to be completed to test each feature or story.

Ex: Create a/c : Identify scenarios

Write test cases

Review test cases

Execute test cases

Defect tracking

II. Daily Stand-Up Meeting/ Role Call Meeting/ Daily Scrum Meeting:

a. Here the entire scrum team meets.

b. This meeting is completely driven by the scrum master.

c. Here every engineer should explain:

- What they have done yesterday?
- What were the impediments/hurdles they faced yesterday?
- What are the Activities he is planning to do today?
- What are the impediments he is expecting in order to complete today's task?

d. The scrum master tries to solve certain impediments right there in the meeting. If it takes too much time then scrum master notes it down in 'Impediment backlog' and solves it later.

e. Generally, this meeting should be completed within 10-15 mins.

f. This meeting should be conducted at the beginning of the day.

g. Here everybody should stand-up in the meeting so that people only talk to the point.

III. Sprint Review Meeting:

- a. Sprint review meeting should be done at the end of the sprint, where the engineers will give a demo to the product owner.
- b. They will also discuss how to plan for the next sprint.

IV. Retrospective Meeting

- a. Here the entire scrum team meets and discusses all achievements (good process followed) and mistakes (wrong activities performed) and it will be documented. This document is called as Retrospect document.
- b. When the next sprint starts while doing sprint planning meeting, we refer this document & we plan it in such a way that old mistakes should not be repeated and good activities are once again adopted.

SOFTWARE TESTING

- The process of finding defects in the software is called as Software Testing.
- Or
- Verifying the functionality of an application against requirement specification is called Software Testing.

Software Testing Principles:

-Software testing follows **7 fundamental principles** that help ensure efficient and effective testing. These principles guide testers in detecting defects and improving software quality. Let's understand each one in simple words.

	Principle	Meaning	Purpose
1.	Testing Shows Presence of Defects.	Testing finds bugs but can't prove there are none.	Focus on finding defects.
2.	Exhaustive Testing is Impossible	Testing every scenario is not possible.	Use efficient test techniques.
3.	Early Testing is Important	Start testing early in development.	Reduce cost and effort of fixing defects.
4.	Defect Clustering	Most defects are in a few modules.	Prioritize high-risk areas.
5.	Pesticide Paradox	Repeating the same tests becomes ineffective.	Regularly update test cases.
6.	Testing is Context-Dependent	Different projects need different testing approaches.	Choose the right testing strategy
7.	Absence of Errors is a Fallacy	Bug-free software can still fail if it doesn't meet needs.	Ensure software fulfills business requirements.

Why should be do Software Testing?

ANS :

- 1) Every software is developed to support some business, if there is any defect in the software, it will effect on the business, so before we use the software for business all the problems should be found and fixed.
- 2) To check whether the software is developed according to customer requirement.
- 3) To improve the quality of the software.

Software Testing can be done in two ways :

- 1)Manual Testing 2)Automation Testing

1)Manual Testing:

- Testing the application again and again manually in order to find defects in the software is called as Manual Testing.
- To do manual testing no programming knowledge is required, No Tools are required.

2)Automation Testing:

- Here Test Engineer will write the script/program/code by using tool called as selenium, and they will run the program against the software, Tool will automatically test the software and gives the result as Pass or Fail.

There are 3 types of Software Testing:

- 1)White-Box Testing
2)Black-Box Testing
3)Grey-Box Testing

1)White-Box Testing/Open Box Testing/ Transperent Testing/Glass Box Testing/ Unit Testing :

- It is done by developers.
- Testing each and every line of the source code is called White Box Testing.

2)Black- Box Testing/Closed-Box Testing/Functional Testing/Behavioural Testing

- It is done by Test Engineers.
- Verifying the functionality of an application against requirement specification is called as Black-Box Testing.

Difference between WBT & BBT

	WBT	BBT
1.	Done by Developer.	Done by Test Engineer.
2.	Require knowledge of internal code.	Not require knowledge of internal code.
3.	Focuses on internal structure of the software.	Focuses on External Behaviour of software.
4.	It is based on Code structure.	It is Based on Requirement Specification.
5.	Can be more complex and time consuming due to its detailed nature.	It is generally simpler and quicker as it doesn't require code knowledge.
6.	It is good for finding the hidden errors and internal flaws.	It is effective at finding functional, Usability, and interface issues.

White-Box Testing/Open-Box Testing/Glass-Box Testing/Transparent Testing/Unit Testing:

- WBT done by developer.
- Testing each and line of source code is called is WBT.

Open-Box Testing: since the source code is visible to the developer, it is called Open-Box Testing.

Unit Testing: Smallest unit of software is one single line of the code, here developers will be checking each and every line of the source code, so it is called as Unit Testing.

Types of WBT:

- 1)Path Testing
- 2)Loop Testing
- 3)Conditional Testing
- 4)White-Box Testing from Memory point of view.
- 5)White-Box Testing from Performance point of view.

1)Path Testing:

-Here developers will write the flowchart / flow-graph, and test each and every individual path.

Advantage of Path Testing:

- 1)We will not miss any path to test.
- 2)We will not test the same path again and again.

2) Conditional Testing:

-Here developers will check for logical conditions, that is for both True and False conditions.

-E.g - if(condition)

```
{
    True
}
else
{
    False
}
```

3) Loop Testing:

-Here developers will check the loop, they will check whether loop is repeating for defined number of the code will increase.

4) WBT Memory point of view:

-Typically mistake done by the developer, Bcz of which size of the code will increase.

-Bcz of Logical mistake.

-Bcz of not using inbuilt function.

-Bcz of repeating the same code instead of using function.

-Bcz of unused variables, and functions.

5) WBT Performance:

-Typically mistake done by the developer Bcz of which it will take more time to execute the program.

-Bcz of not using by better logic.

BLACK-BOX TESTING

-Also known as Functional Testing/Closed-Box Testing/Behavioural Testing.

-It is done by the Test Engineer.

-Verifying the functionality of an application against requirement specialization is known as BBT.

There are basically 12 types of BBT:

- 1) Functionality Testing
- 2) Integration Testing
- 3) System Testing
- 4) Acceptance Testing
- 5) Adhoc Testing
- 6) Smoke Testing
- 7) Usability Testing
- 8) Compatibility Testing
- 9) Performance Testing
- 10) Exploratory Testing
- 11) Globalization Testing
- 12) Regression Testing

Closed-Box Testing:

-Since, the source code is not visible to the test engineer, so it is called as Closed-Box Testing.

1) Functionality Testing/Component Testing/Field Level Testing:

-Testing each and every components of application thoroughly against requirements specification.

Components:

- ✓ Text Field ✓ Drop-down Box ✓ Radio Button ✓ Check Box ✓ Button (Submit, Reset, etc.)
- ✓ Hyperlink ✓ File Upload ✓ Date Picker ✓ Search Box ✓ Pagination ✓ Table/Grid View
- ✓ Toast Messages/Notifications ✓ Navigation Menu ✓ Tabs ✓ Sliders/Carousels
- ✓ Tooltip
- ✓ Toggle Switch ✓ Modal/Popup Window ✓ Captcha Verification

Thoroughly : Testing each and every component of an application by entering positive and negative input is called as thoroughly.

Types Of Functionality Testing?

1)Over Testing 2)Under Testing 3)Optimized Testing

1. Over Testing: Testing the application with same set of scenarios in different way is called as Over Testing.

Drawback of Over Testing : Waste of Time.

2. Under Testing: Testing the application with insufficient set of scenarios is called as Under Testing.

Drawback of Under Testing: By doing under testing we will miss lot of scenarios.

3. Optimized Testing: Testing the application with those scenarios which make some sense is called as Optimized Testing.

Advantage of Optimized Testing: We will not waste our time and any scenario.

How to do functionality?

ANS : Steps to perform Functionality Testing:

1. Understand the Component Requirements. Eg. input for that component.
2. Prepare Test Cases for the Component. (Cover positive, negative, boundary scenarios)

✦ **Example:** If testing a **Login Module**, test cases may include:

Valid username & password → User logs in successfully. Invalid password → Error message displayed.

Empty fields → Validation message appears.

3. Execute test cases and find bugs.

Functionality Test Scenario:

1. Component name: **Register Button.**
 - To check register button is enabled.
 - To check when user click on register button registration page should be displayed.
 - To check when user place the cursor on the register button, cursor should be changed to the pointer.
 - To check that, when user clicks on register button then next page is opened in same tab or not.

2. Component name: **Username text field.** : It should accept 5-7 alphabets.

Req	Component Name	Input	Expected Output
1.1	Username Text-field	abcde	Accept
		abcdef	Accept
		abcdefg	Accept
		ABCDE	Accept
		AbCdEfG	Accept
		12345	Reject
		!342@5	Reject

Why requirements should be in

number? Ans: 1)Requirement

becomes measurable.

2)Requirements becomes traceable.

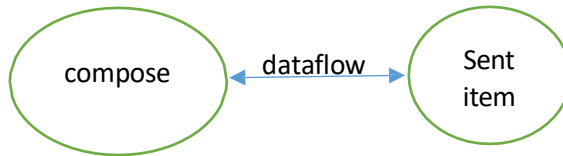
3)There will be clarity in the requirement.

4)There will be good communication between developers and test engineers.

2) Integration Testing/ Dataflow Testing/ Interface Testing:

- Testing the dataflow between or interface between the 2 or more modules is called integration testing.

Eg:



Application : **Watsapp**

Scenarios:

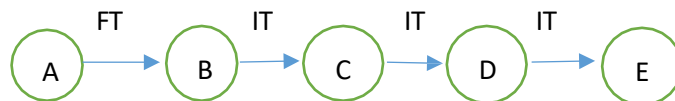
1. Login as user A, send message to user B. Then Logout as user A. Then Login as user B and check whether the message is received or not.
2. Call user A to User B and Check whether the call is received or not to user B.
3. Etc.

Types of Integration Testing:

- 1) Incremental Integration Testing
- 2) Non-Incremental Integration Testing

1) Incremental Integration Testing:

Incrementally add the modules and check the dataflow between the modules. Eg .



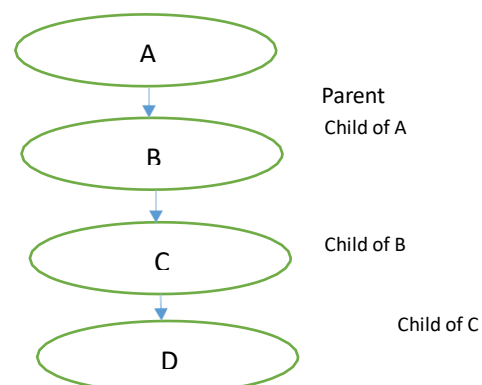
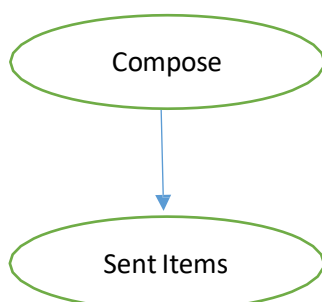
There are two types of Incremental Integration Testing:

- A) Top-Down Incremental Integration Testing
- B) Bottom-up Incremental Integration Testing

A) Top-Down Incremental Integration testing:

- Incrementally add the modules and check the dataflow between the modules, and make sure the module which we are adding should be the child of previous module.

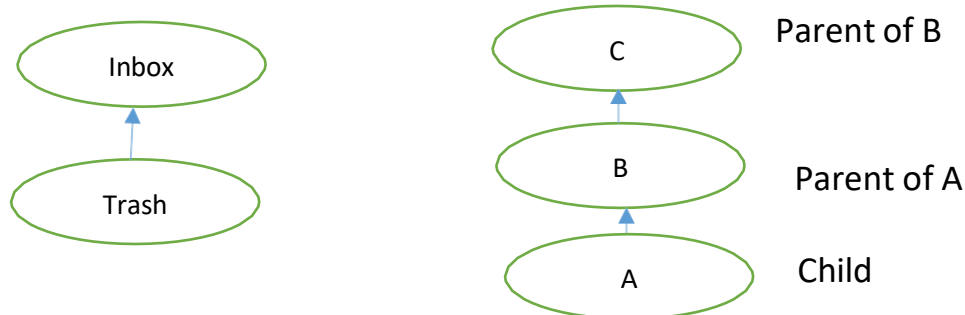
Eg :



b) Bottom – up Incremental Integration Testing:

-Incrementally add the modules and check the dataflow between the modules and make sure that the module which we are adding should be the parent of previous module.

Eg.



2) Non – Incremental Integration Testing:

- Whenever we don't know which is parent module and which is child module,
- When there is N number of dataflow at that time we go for Non- Incremental Integration testing.
- To do this type of testing, Combine all the modules at once and check the dataflow between the modules.

Drawbacks of Non- Incremental Integration Testing:

- 1) Chances are there we might end – up testing the same dataflow again and again.
- 2) Chances are there we might miss to test some dataflow.
- 3) If there is defect in a particular module it is difficult to identify where exactly the defect is.

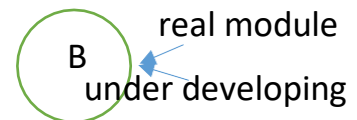
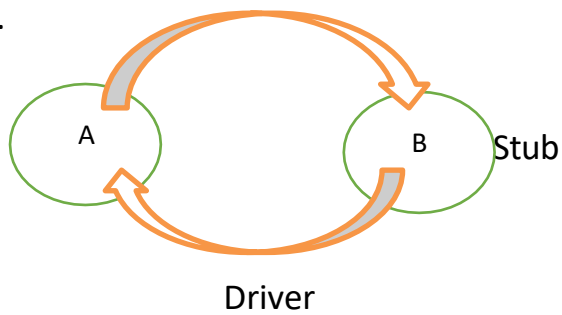
Stubs and Drivers:**a) Stubs:**

- Stubs is a dummy module, which acts like a real module, which is not yet developed.

b) Drivers:

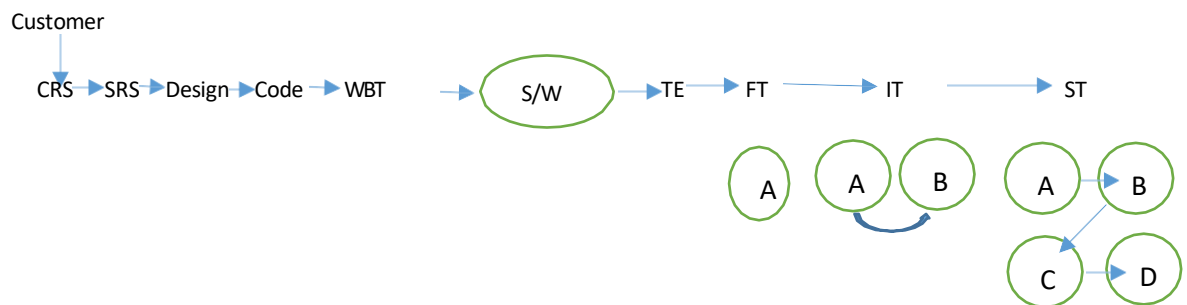
- Drivers creates a media or interface to communication between the real mode and stubs for Integration Testing.

Eg.



3) System Testing:

- It is End to End testing, where in test environment just similar to production environment.
- **End to End Testing:** Navigate through all features, and check whether the end feature/module is working as expected or not.



Eg:

Scenario : Flip cart ->My Orders(Module)

1)open flip cart application. 2)search the product. 3)select the product and add into wish list. 4)Go to wish list and add the product to cart. 5)click on Buy now button. 6)Select payment option. 7)add address and click on buy button. 8)Go to My orders and check whether the order is placed.

Types Of Environment:

1)Development Environment 2)Testing Environment 3)Production Environment 4)Pre-Production Server.

1)Development Environment:

- It is a set up where all the Development activities will take place.
- It consist of Hardware, Software, Server, Network.

2)Test Environment:

- It is a setup where all the Testing activities will take place.
- It consist of Hardware, Software, Server, Network.

3)Production Environment:

- It is a setup where all the business activities will take place.
- It consist of Hardware, Software, Network, Server.

4)Pre- Production Environment:

- It is a Test Environment which is just similar to production environment.

When to go for System Testing:

- 1) When the bunch of modules are ready.
- 2) When the product is functionally stable. (Most of the defects found and fixed).
- 3) When there is a both Test Environment is just similar to Production Environment.

4)Acceptance Testing:

- It is End to End Testing done by IT Engineer, sitting in customer place referring to the real time business scenarios and check whether the software is capable to handling it or not.
- Acceptance testing ensures that a software application meets business requirements and is ready for release.
- It is the final phase of testing before deployment.

Why customer will do Acceptance Testing?

- Under the business pressure Software Company might push the software with the defects.
- If the software is launched with the critical defects, the customer might undergo huge loss.
- Chances are there requirement and develop the wrong software that is why customer will do Acceptance Testing.

Approaches of Acceptance Testing:

Approach	Who Tests?	Purpose	Example
User Acceptance Testing (UAT)	End users	Ensures the software meets user needs and expectations	A bank app is tested by customers to check if they can transfer money easily.
Business Acceptance Testing (BAT)	Business analysts, stakeholders	Confirmsthat the software aligns with business goals	A shopping website owner checks if orders and payments work smoothly.
Contract Acceptance Testing (CAT)	Customers & development team	Verifies that the software follows the agreed contract terms	A company ensures its new HR system has all promised features like payroll and leave tracking.
Regulation Acceptance Testing (RAT) / Compliance Testing	Legal teams, compliance officers	Ensures the software follows industry regulations and security guidelines	A banking app is tested to confirm it follows RBI security standards.
Operational Acceptance Testing (OAT) / Production Readiness Testing	IT operations, system administrators	Checks system stability, performance, and integration in a live environment	A food delivery app is tested to ensure it runs smoothly with high traffic
Alpha Testing	Internal testers	Identifies bugs before release	A gaming company tests a new game internally for crashes and gameplay issues.
Beta Testing	Real users	Collects feedback from real users before final launch	WhatsApp releases a beta version of a new feature for limited users to test.

5)Smoke Testing/ Dry Run Testing/ Health Checkup of a Product/ Building Verification Testing

- Testing the Basic and Critical features of an application is called as Smoke Testing.
- Smoke testing is a **quick and basic test** to check whether the **main functions of software work properly** before doing detailed testing.
- It helps to **catch major bugs early** and ensures that the build is stable enough for further testing.

Ex:

Imagine you are testing an **e-commerce website**. After a new update, you perform smoke testing by checking:

- If the **homepage loads** properly.
- If users can **log in** successfully.
- If products are **displayed** on the product page.
- If the **basic functions don't work, there's no point in**

testing further Advantages of Smoke Testing:

- 1) Test engineer can find the defects in early stage.
- 2) Developers get sufficient time to fix defects.
- 3) Testing team saves time and effort.
- 4) Ensures the build is stable for further testing.
- 5) Avoids wasting resources on faulty builds.
- 6) Speeds up the overall testing process.
- 7) Helps in delivering quality software faster.

How to do Smoke Testing?

Step 1: List all

Modules. Step 2:

Make two Columns.

- a) Features need to tested.
- b) Features need not to be tested.

Step 3: write

Scenario. Eg:

Application name :

Youtube Step 1: List all

Modules

Video, Setting, Gmail account, Login, Logout, Search Bar, saved, History, Like, Comment, Share, Subscribe.

Step 2: Make two columns

Features Need to be Tested	Features Need not be Tested
Video Content Login Logo ut Save d Search Bar	History Gmail account Like Comment Bell icon

Step 3: Scenario

Login: When user clicks on Login button, to user should be displayed on Homepage of you tube.

Logout: When user clicks on Logout out user should be go outside from you tube/ user can't see the content on you tube.

Search Bar: When user clicks on search bar, user can able to search what he want.

When to do Smoke Testing?

- 1) Developer before they give the software to the test engineer they should do smoke testing.
- 2) Whenever Test Engineer gets a new build from development team they should do smoke testing.
- 3) Customer, Before He does Acceptance they should do smoke testing, because to check whether he has received complete product and to check whether software is properly installed or not.
- 4) Once the software is involving in installing the software, they should do smoke testing to check software is properly installed or not.

Why we do smoke Testing?

- To detect major defects early and prevent deeper testing on faulty software.
- To ensure the build is stable before moving to detailed testing
- To save time and effort by stopping testing if core functions fail.
- To help developers fix defects quickly with early feedback.
- To speed up the testing process by verifying key functionalities fast.
- To improve software quality by ensuring critical features work properly.
- If basic functions fail, stop testing and fix them first!

Difference between Smoke Testing and Sanity Testing:

Smoke Testing Example: Before detailed testing, I check if the login, product search, and checkout functions work in an e-commerce app. If any fail, I stop testing and send it back for fixes.

Sanity Testing Example: After fixing a checkout issue, I test only the checkout page to ensure the bug is resolved and nothing else is affected.

- Smoke = Basic Check with only Positive Testing (Build is stable or not?)
- Sanity = Quick Recheck with Positive and Negative Testing (Bug fix is working or not?)

	Smoke Testing	Sanity Testing
1	Done on a new software build before detailed testing.	Done after bug fixes or minor changes to verify the fix.
2	Checks if basic functionalities work properly .	Verifies if a specific issue or feature is working correctly .
3	Covers all major features of the application.	Focuses only on the affected module or feature .
4	Quick and broad check to ensure the build is stable.	Narrow and focused check to confirm the fix.
5	Example: In an e-commerce app , test login, product search, and checkout.	Example: After fixing a payment issue , test only the payment module.
6	If it fails, the entire build is rejected , and testing stops.	If it fails, only that module needs rework , but testing continues on other parts.
7	It is done By both Test engineer and Developer .	It is done by only Test Engineer .
8	Here we do Positive Testing .	Here we do Positive and Negative Testing .

6)Adhoc Testing/ Monkey Testing/ Gorilla Testing

- Testing the application randomly in order to find defects in software, without looking into requirement specification is called as Adhoc Testing.
- Think like a user, break like a tester!

Example 1:

Suppose you are testing a **food delivery app**. Instead of following test cases, you randomly:

Place an order without selecting a restaurant.

Enter emoji or special characters in the address field.

Quickly switch between cart and payment pages multiple times.

If the app crashes or behaves unexpectedly, you have found a defect through adhoc testing.

Example 2: InstaPro

We can OFF or ON msg seen.

We can save any video, post, reel.

Why we should do Adhoc Testing?

- Chances are there end user might use the application randomly and find the defect.
- Developers will develop the software by looking into the requirement, similarly Test engineer will test the software by looking into the requirement chances of finding defects will be less.
- Test engineer intension of doing adhoc testing is to break the product.
- Find hidden bugs that structured testing might miss.
- Helps catch crashes and performance issues early

When to go for Adhoc Testing?

- 1) When the product is functionally stable.
- 2) Whenever test engineer is free, they should do adhoc testing.
- 3) After formal testing to find extra hidden bugs.
- 4) When there is limited time, and quick testing is needed.
- 5) Before release to ensure no last-minute surprises.
- 6) When a critical bug is fixed, to check for unexpected side effects.
- 7) To test real-world user behavior, beyond scripted test cases.

Volume Testing: Testing the stability and response time of an application by transferring huge volume of data is called as Volume Testing.

Soak Testing: Testing the stability and response time of an application by applying load for particular period of time is called as Soak Testing.

8)Globalization Testing

- If the application is developed for multiple language, then it is called as Globalization.
- Testing that application which is developed for multiple language is called as Globalization Testing.
- **There are two types of Globalization Testing**
 - Internationalization (I18N) 2) Localization (L10N)

1) Internationalization (I18N) Testing:

- Testing the application which is developed for multiple languages is called as internationalization Testing.
- Here we check
 - 1) Language is correct or not.
 - 2) Content is right place or not.
 - 3) We check whether page is broken or not.

How to do Internationalization Testing?

- 1) As a Test Engineer I will talk to developer and go to property file and add prefix and suffix to the content.
- 2) Save the property file.
- 3) Open the software and change language.
- 4) I will check for prefix, if prefix is correct means Language is correct.
- 5) If suffix is correct means Content is correct.

2) Localization Testing (L10N):

- Testing the application locally and checking whether it is changing according to country standard is called as Localization Testing.
- Here we check country currency, flag, date and time.

9)Exploratory Testing

- Understand the application, understand how each and every feature will work, identify the scenario, Prioritized scenario, Document the scenario.
- Or
- Explore the application and understand each and every modules in depth and identifying the scenario as per your understanding is called as Exploratory Testing.

When to go for Exploratory Testing?

- When there is no requirement or if the requirement is missing at that time we will do exploratory testing.

How to do Exploratory Testing?

- Explore the application and understand the application.
- Enter positive and negative inputs.
- Explore the application and check the dataflow between the modules.
- Explore the application and navigate through all the features and check that end feature is working as expected.

Drawbacks of Exploratory Testing:

- 1) Chances are there we might miss-understand the defect as a feature.
- 2) Chances are there we might miss-understand feature as a defect.
- 3) If the feature is really missing, we will not get to know.
- 4) It is time consuming.

How to overcome these Drawbacks?

- By using some common sense.
- By product Knowledge.
- By Domain Knowledge.

10)Usability Testing

- Testing the user friendliness of an application is called as Usability Testing.

How to do Usability Testing?

- Here we check Look and Feel of an application.
- Here we check how easy the application is to understand.
 - Whatever the application that has been developed, it should be available to the customer within 3 clicks.

When to do Usability Testing?

- When the application is functionally stable.

What kind of application we do Usability Testing?

- The application which is used by variety of users.
- The application which generates huge revenue.
- The application wherein don't provide any training to the end users.

11)Compatibility Testing

- Testing the application of different hardware and software configurations and browsers is called as Compatibility testing.
- Compatibility testing ensures that a software application works smoothly across different devices, browsers, operating systems, networks, and hardware configurations without issues.

Why we do Compatibility Testing?

- Ensures a **consistent user experience** on all platforms.
- Identifies issues related to **UI, functionality, or performance** on different environments.
- Helps **avoid customer complaints** due to compatibility issues.

How to do Compatibility Testing?

- 1) Identify the **target environments** (devices, browsers, OS, etc.).
- 2) Set up a **test matrix** to cover all combinations.
- 3) Perform **manual or automated testing** on each platform.
- 4) Compare results and **log defects** for inconsistencies.
- 5) Verify fixes and **ensure cross-platform consistency**.

To do compatibility testing Use cloud-based tools like Browser Stack or Source lab.

12)Regression Testing

Testing the unchanged features to make sure that is not affected or broken because of changes is called as Regression testing (here changes can be addition, modification, removal of features or bug fixes) (OR)

Re-execution of same test case is different test cycle/ sprint/ release to make sure that changes are not introducing any defects in unchanged features (Changes can be addition, modification or removal of features) is called Regression testing

Type of Regression testing:

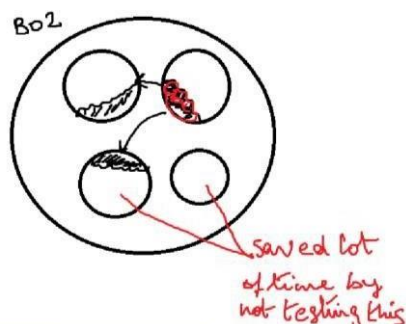
- 1) Unit regression testing
- 2) Regional regression testing
- 3) Full regression testing

a) Unit regression testing:

Testing the changes (or) only the bugs which is fixed is called as unit regression testing.

b) Regional regression testing:

Testing the changes & only the impacted regions is called regional regression testing



How will you identify impacted Region?

1. Based on product knowledge

As a TE in-depth I will be knowing how each & every module works & also I will be knowing how all the modules are related based on that knowledge, I will be able to identify impacted areas.

2. By preparing Impact matrix

Here we list the changes & also all the features in the application, then mark the impacted areas.

3. By conducting Impact analysis meeting

As soon as the new build comes entire testing team meets & discuss about list of bugs fixed & the impacted areas

4. By interacting with customer, business analyst, development team, testing team we should gather the impacted areas & create an impact list, based on that we should do Regression testing

Advantages of Regional Regression Testing:

1. By not testing certain features we are saving testing time which intern reduces the testing cost
2. Test cycle duration Reduces because of that turnaround time taken to deliver the product to the customer reduces

Dis-advantages of Regional Regression Testing:

1. Chances are there we might miss identifying the impacted area because of that we might miss the bugs.

c) Full Regression testing:

Testing the changes and all the remaining features is called as Full Regression testing

Why/when we do full regression testing?

1. Whenever too many changes are done in the product better to do full regression testing

2. If the changes are done in core features
3. Every few (4-5) cycles once we should do full regression testing & last few cycles we should do full regression testing because we are about to launch the product to the production to not to take any risk.

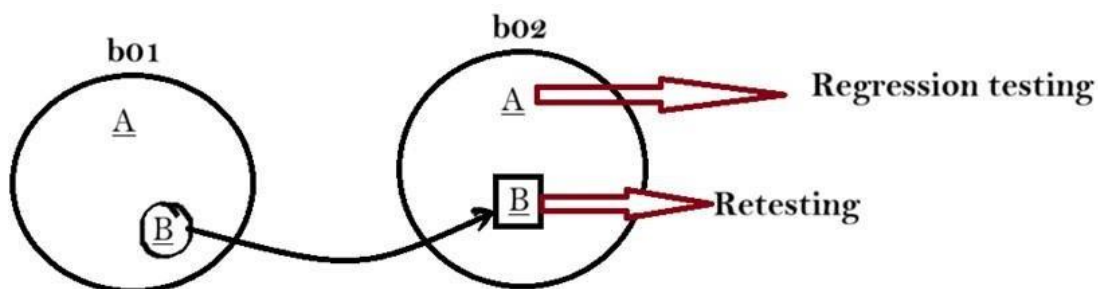
What is the difference between Regression Testing and Re-testing?

<u>Regression Testing</u>	<u>Retesting</u>
Fixing the bugs or doing changes might have impact on other features so testing the unchanged feature to make sure that it is not broken because of changes is called as Regression Testing	Whenever developer gives build, checking or verifying whether defect is fixed or not is called Retesting.
Regression Testing is done for passed test cases	Retesting is done for failed test cases
We go for automation	We don't go for automation

Progression testing.

When we do regression testing?

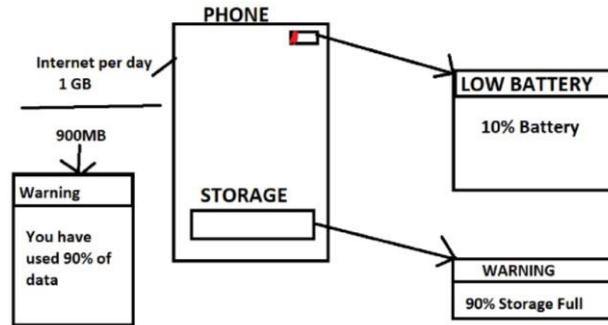
- 1st release 2nd build onwards we do regression testing
- 2nd release 1st build onwards we do regression testing



NON – FUNCTIONAL TESTING

Yellow Box Testing:

-Testing the warning message of an application is called as yellow box testing.



Comparison Testing:

-Testing the newly build application with the similar kind of application which is released in the market, here we compare the application, check the advantages and dis-advantages and check whether all the features are present in our newly built application or not.

Ex: Zomato with Swiggy
Amazon with Flipkart
Ola with Uber

Non-Functional Testing

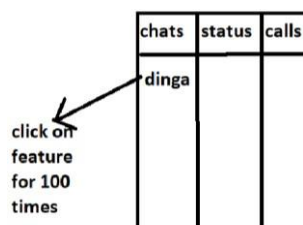
Accessibility Testing:

-Testing the user friendliness of an application from physically challenged people point of view.

Ex: We check just by entering keyword whether we are able to access the objects
ATM machine for blind people

Reliability Testing:

-Testing the functionality of an application continuously for a particular period of time is called as reliability testing.

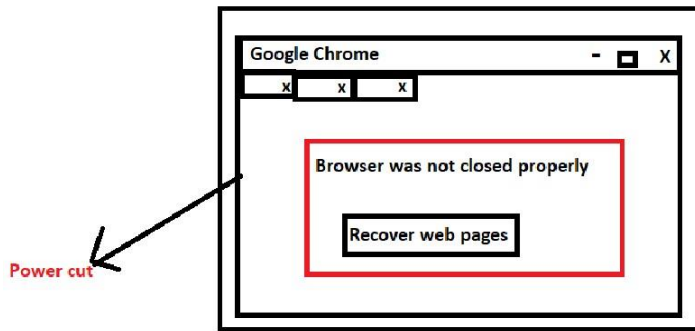


Automation

```
while(200)
{
    click on chats;
}
```

Recovery Testing:

-Testing the functionality of an application to check well the application recovers the data from the crash or disasters.



Compliance Testing:

-It is a type of non-functional testing which is done to check the sw that that is developed meets the company standard organization respective standards or not.

Pilot Testing:

-It is done by set of peoples, wherein they do trail run on project, give feedback about project to company before Release.

Risk Based Testing:

-It is testing done by test engineer for the project based on the risk. Here test engineer will test feature or functionality.

Migration Testing:

Whenever the technology is outdated or when database is outdated that time we will migrate from old technology to new technology or from old database to latest database, in that case we go for migration testing.

Interrupt Testing:

-It is a process of replicating the interrupt or process of replicating abrupt interrupt in application or software is called interrupt testing.

System Integration Testing:

-Testing the dataflow between 2 different system or software or application and also make sure that end to end business flow is working fine between 2 system or application as expected Is called as System integration testing.

Eg:

WhatsApp >> invite friends >> fb messenger

TYPES OF TESTING

What are the types of software testing ?

1st answer :

1) BBT 2) WBT 3) Grey box testing

2nd answer :

1) Functional Testing 2) Non-Functional Testing

1) **Functional Testing:** FT , IT, ST, ADHOC TESTING, SMOKE TESTING, REGRESSION TESTING, EXPLORATORY TESTING, ALPHA and BETA TESTING, ACCEPTANCE TESTING,

2) **Non- Functional Testing:** USABILITY TESTING , PERFORMANCE TESTING, RECOVERY TESTING, RELIABILITY TESTING, GUI TESTING , YELLOW BO GLOBALIZATION TESTING, COMPARISON TESTING ,COMPATIBILITY TESTING , MIGRATION TESTING ,WEB SECURITY,

3rd answer :

A) Dynamic testing B) static testing :

A) Static testing :

- 1) Static testing involves all the verification activities.
- 2) Verification activities involves reviews, walk through, inspections, and auditing.
- 3) Here we try to prevent the defects.
- 4) Static testing is done before software is developed
- 5) In static testing we ensure that are we building product right 6)it is less cost.
- 7) Here we don't execute code.

B) Dynamic testing :

- 1) Dynamic testing involved all validation activities.
- 2) Validation includes actual testing such as FT, IT, ST, smoke, adhoc testing etc....
- 3) Here we try to find the defects.
- 4) Dynamic testing is done after software is developed.
- 5) In dynamic testing we ensure that are building right product.
- 6) It is more cost
- 7) Here we execute the code.

4th answer :**A) QA(Quality assurance) or Process:**

- 1) QA is the set of activities which ensures quality in process
- 2) This is pro-active process
- 3) Aims to prevent the defect
- 4) Verification is best example for QA
- 5) QA is process oriented
- 6) Goal of QA is to improve development and testing process so that defect should not rise in future
- 7) QA is planning
- 8) Here we identify weakness in process and try to improve

B) QC(Quality check) or product :

- 1) QC is the set of activities which ensures quality in product
- 2) This is reactive process
- 3) Aims to find the defect
- 4) Validation is best example for QC
- 5) QC is product oriented
- 6) Goal of QC is to find defect in software while testing.
- 7) QC is executing the plan
- 8) We try to identify defect in s/w and improve quality in product.

5TH answer:**a) Formal testing :**

- Testing the software by following the procedures or guidelines or documents is called as formal Testing.
- He will Document all the test plan, test scenario, test case and review all document.

b) Informal Testing:

- Testing the software without following the procedures or guidelines or documents is called as Informal testing. --- Here we will not have document for test plan, test cases, test scenarios.

c) Accessibility Testing :(ADA testing > American disability act)

- Testing the application or software from physically challenged people point of view is called as Accessibility Testing.
- Ex : ATM testing for deaf and dumb people.

Thick & Thin Client: (Standalone application or client server application)**1) Thick Client:**

- Here the software will be installed in client place or client machine and this software is called as Thick Client.
- Ex: Word, Notepad, Whatsapp, Chrome Browser, Facebook.

2) Thin Client: (Web Application)

- Here the software will be installed in server and this software will be used to do work, this is called Thin Client.
- Ex: www.gmail.com, www.facebook.com etc.

TEST CASE

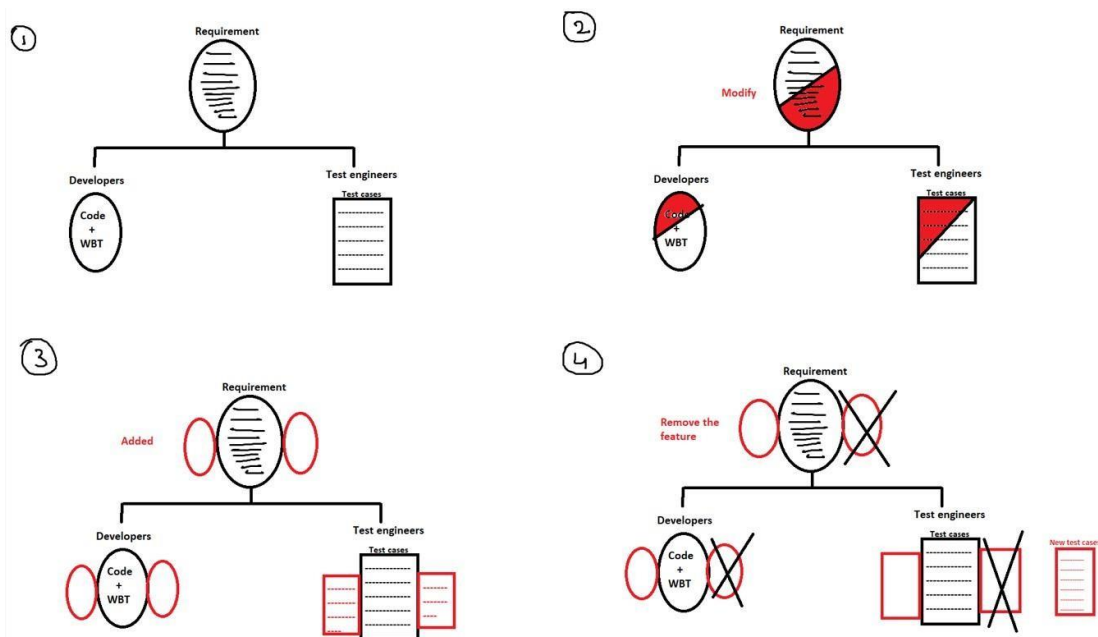
Test case is a document that covers all possible scenarios for a specific requirement. It contains different sections like Step number, Input, Expected result, Actual result, Status, and Comments.

What is the drawback of seeing the requirements and testing the software?

1. There will be no consistency in testing if you look into requirements and test the software.
2. Quality of testing varies from person to person
3. Quality of testing depends on the memory power of the test engineer.
4. Quality of testing depends on the mood of the T.E.

When do we write test case?

1. When developers are busy in building the product, the testing team will be busy writing the test cases.
2. When the developers will modify or change the feature, parallelly T.E. will modify or change the test cases.
3. When the developers will add the features parallelly test engineers will add new test cases.
4. When the customer is removing the requirement, developers will remove the feature, and parallelly test engineer will remove the test cases to make sure that features are removed from the s/w or not.



Why do we write Test cases?

1. We write test cases to have better test coverage.
 - a. When the requirement comes in developers are busy building the product same time test engineers are free, so they identify all possible scenarios and document it.
 - b. When the build comes we can spend time executing the scenarios, because of this no. of scenarios that you are covering will be more.
 2. To have consistency in test execution.
 - a. It means if you have documented the scenarios, you can make sure that you are executing all the scenarios in all the test cycles, sprints, or releases.
 3. To depend on the process rather than person.
 4. To avoid training every new engineer on the product or on the requirement.
 5. Test case is the only document that acts like as proof for customer, development team, and also manager that we have covered all possible scenarios.
 6. Test case acts like a base document for writing automation scripts, if you refer to the test case and write automation scripts you can ensure the same kind of coverage even in automation.
 7. If you documented the test case, no need to remember the scenarios.
 8. If you have documented the test cases, test execution happens in a very organized way.
 9. If you have documented the test cases, the time taken to execute is much less.
- (4-5 points is enough)

Difference between test case & test scenarios:

Sr.	Test Scenario	Test Case
1	It is a high-level document of all the customer business workflow according to the customer's requirement.	It is a detailed document of the scenario that helps us to test the application.
2	We write Test scenarios by looking into the requirements.	We write test cases by looking into both requirement and test scenarios.
3	By looking into test scenarios, we can't test any application until you have good product knowledge.	We can test any application by looking at the test case, no matter if you have product knowledge or not.
4	Here we mention what to test.	Here we mention how to test.

TEST CASE TEMPLATE

- Every TE will write test case in Test case Template only
- Test Case can be prepared in “Test Case Management Tool” or “MS Excel”
- ➤ Test case template is divided into 3 sections:
 1. Header
 2. Body
 3. Footer

How to fill the header?

1. Testid:

Format: test case name _number Ex: FT_101

2. Requirement Number:

BA when he converts CRS to SRS, in SRS for each requirement he will write the requirement no.

Ex: 30.1 Amount Transfer

30.1.1 FAN text field

30.1.2 TAN text field

30.1.3 Amount text field

3. Test data:

It is the data written by a TE and has to be done before the test execution.

Ex: TE should have UN, PWD, URL, a/c number

4. Pre-condition:

It is a set of actions or settings which should be ready/done by TE before executing the 1st test case.

Ex: User should have balance in his account.

5. Severity:

TE will give severity for every individual test case, based on how important and complex the feature is from customer's point of view. TE will execute test case based on severity.

There are 3 types of severity for Test cases: Critical/ Major/ Minor

Features	Test case	Severity	Order of execution
Login	TC 1	Critical	TC 1
Drafts	TC 2	Major	TC 5
Inbox	TC 3	Critical	TC 3
Trash	TC 4	Minor	TC 2
Compose	TC 5	Critical	TC 4

6. *Test case type:*

Here the TE mention what type of test case he is writing.

Ex: Functionality test case, Integration test case, system test case.

7. *Brief Description:*

It describes about the complete test case and the behavior of the test case. Ex: In the amount transfer module, it should accept only +ve integers.

How to fill the footer?

1. Author: Person who writes the test case Ex: Dinga
2. Reviewer: Person who reviews the test case Ex:
Dingi
3. Approved by: Person who approves the test case
Ex: Test lead
4. Approval date:
Ex: 01-01-2023

TEST CASE DESIGN TECHNIQUES

It is a technique which is used while writing test case in order to improve test coverage.

Types of Test case design techniques:

1. Error guessing
2. Equivalence class partitioning
3. Boundary value analysis (BVA)

1. Error Guessing:

Here we guess all possible errors and we derive the scenarios.

We guess errors based on:

- i. Requirement
- ii. Experience
- iii. Intuition

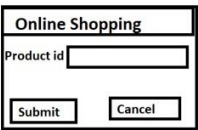
2. Equivalence class partition:

Rule 1 Pressman Rule:

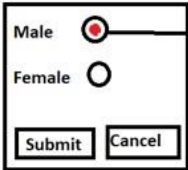
Rule 1: If the input is a range of values, then design test case for one valid and two invalid inputs.

Ex:-
 Req:- 100 - 5000
 Range = [100 - 5000]
 1 valid → 500
 2 invalid → 90, 6000

Rule 2: If the input is in a set of values, then design test case for one valid and two invalid inputs.

Ex:-

 Printer = 10
 Scanner = 20
 Webcam = 30
 Set = [10, 20, 30]
 1 valid → 20
 2 invalid → 25, 40

Rule 3: If the input is in Boolean, then design the test case for both true and false values.

Ex:-

 Select this and test (true)
 unselect this and test (false)
 Select ⇒ true
 deselect ⇒ false

a. Practice Method:

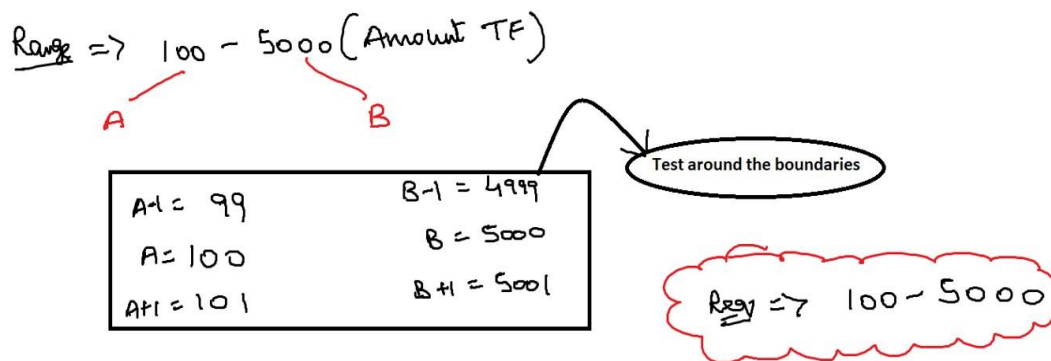
If the input is in range of values then divide the range into equivalent parts and test for all the values, make sure that you are testing for at least two invalid values.

Note:

1. If there is a deviation between the range of values then we go for Practice method.
2. If there is no deviation between the range of values then we go for Pressman rule.
3. By looking into requirements, we will get to know whether there is a deviation or not.

3. Boundary Value Analysis:

If the input is range of values b/w A to B then design test case for A, A+1, A-1 and B, B+1, B-1.



Why we catch more bugs around the boundary?

```
if(Amount > 100 && Amount < 5000)
{
    Transfer
}
else
{
    Throw error msg
}
```

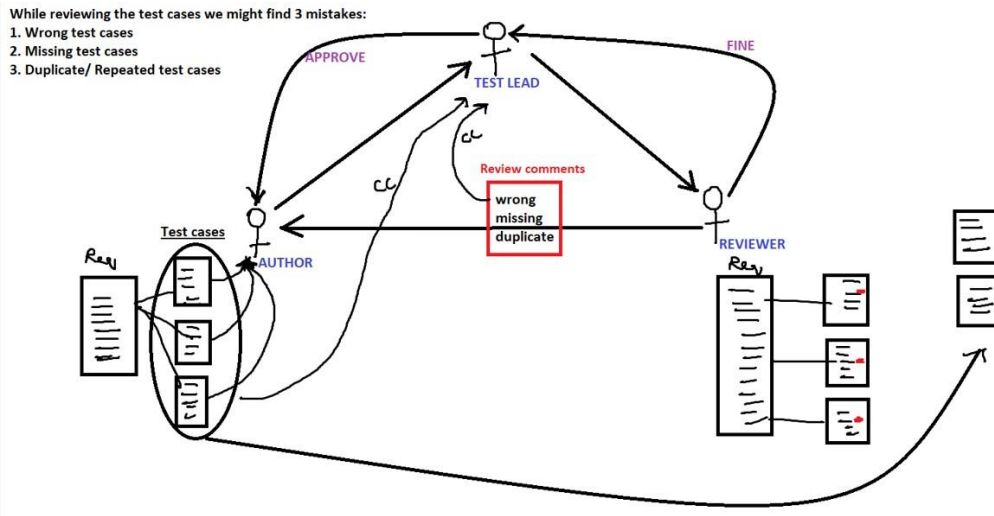


According to this example, developers will do a lot of mistakes while writing code for conditional/ logical operators and that's why chances of finding defects will be more around the boundary.

Test case optimization:

The process of removing the duplicates from the test cases (OR) Removal of repetition of test cases is called test case optimization. We can cover both 1 valid and 2 invalids in BVA itself, so we can skip equivalence partitioning (in only specific cases).

TEST CASE REVIEW PROCESS



On what basis you assign someone to review test case?

They will assign to the person:

1. Who is working on a similar or related module in the project.
2. Who has worked on same module in the previous project.
3. Who has been working in the project since the beginning knows every corner of the product.
4. Who is responsible, and who will understand the requirement very fast and identify more mistakes.

Review ethics:

1. Always review the content, and not the author.
2. Reviewer should spend more time in finding the mistakes rather than giving the solution for it
3. Even after review if there are still any mistakes, both author and reviewer are responsible

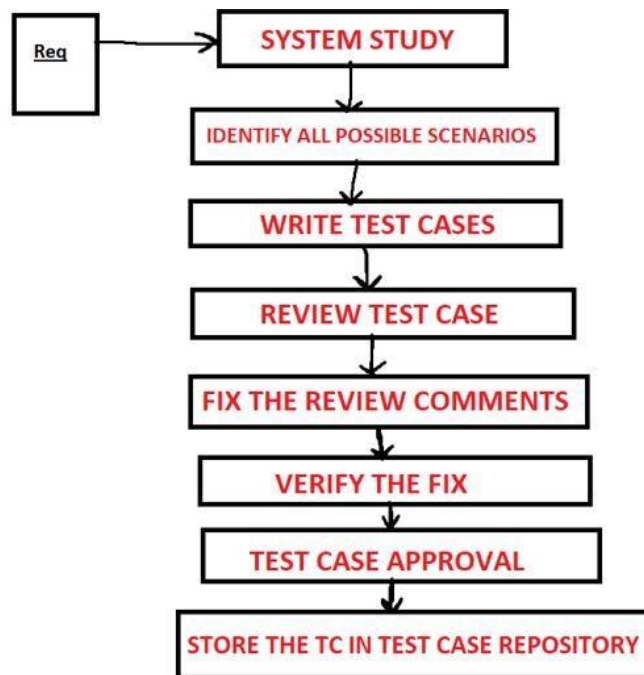
Why we review test case (OR) if I give you the test cases what is the approach to review the test cases (OR) what kind of mistakes will you find while reviewing the test case?

1. I will look into the header of the test case and understand the req for which test case is written and then go to body of the test cases and try to find:
 - a. Missing scenarios b. Wrong scenarios c. Repeated scenarios
2. I will check whether organized or not so that when executed it should take less time
3. I will check whether it is simple to understand so that the it is given to ne TE he should be able to execute without asking any questions
4. I will look into the header of the test case and try to find:
 - a. All the attributes are covered or not
 - b. Check whether content in all the attributes are cover or not
5. I will check whether test case format is according to standers defined in the project

TEST CASE REVIEW DOCUMENT

Sl no.	Test case name	Step no.	Reviewer		Author
			Comments	Severity	
1	CB_AT_Functionality scenarios	8	Transfer more than the balance scenario is missing	Critical	Not fixed (he will give the reason)
		Pre condition	Pre condition is missing	Critical	Fixed
		Header	Test case type is missing	Major	Fixed

PROCEDURE TO WRITE THE TEST CASE



1. System study:

Read the requirement, understand the requirement, and if you have any queries interact with the customer, B.A.

2. Identify all possible scenarios:

- i. Identify
- ii. Brainstorming sessions
 - a. Presentation of feature: Improved product knowledge
 - b. Present scenarios: Missing, Wrong and Repeated scenarios
- iii. Measure the efficiency of Brain storming session

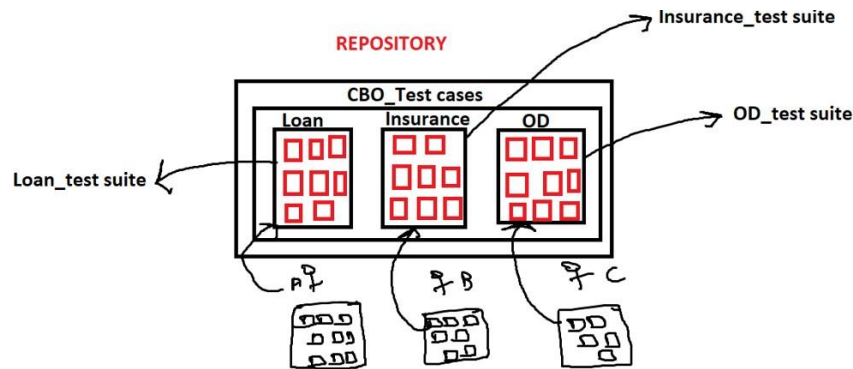
3. Write the test cases:

- Document the identified scenarios □ Group all related scenarios.
- Prioritize the scenarios within each group.
- Apply test case design technique.
- Use standard test case format

4. Store in test case repository:

Test case repository: It is a centralized place where all the test cases are stored.

Test Suite: Collection/ group of related test cases



Q.1 Where do you write the test case?

1. In Excel sheet
2. In Word file
3. In a test case management tool

Ex: Test link, ALM

Q.2 Where do you store the test cases?

1. In a shared folder
2. In a test management tool Ex: Test link, ALM
3. In a version control tool Ex: CVS, Subversion

Q.3 How do you ensure that your test coverage is good? Or How do you convince you customer / management/developer that you have tested everything?

My test coverage is good because my test cases are good. My test cases are good because I have followed a strict procedure Strict procedure means:

1. I did a thorough system study because of which I had a deeper knowledge of the product because of that I was able to find more numbers of scenarios
2. I identified all possible scenarios then I did a brainstorming session because if which I was able to find more numbers of scenarios
3. While writing test case I applied test case design technique because of that there was improvement in the coverage
4. I got all my test cases reviewed and found missing scenarios because of this there was as improvement in the coverage
5. While executing test cases, I identified few new scenarios, I added them back into the test cases because of this there was as improvement in the coverage
6. Also, I performed Adhoc testing because of that there was an improvement in the coverage

I wrote traceability matrix and ensured that every requirement has got at least one test case because of that there was an improvement in the coverage.

SEVERITY & PRIORITY

SEVERITY

- The impact of defect on customer business workflow is called as severity.

There are 4 types of Severity

1) Blocker Defect 2) Critical Defect 3) Major Defect 4) Minor Defect

1) Blocker Defect: Assume that there is defect in the software because of which Test engineer will be blocked to do further testing.

Eg: Blank page, Login button not working, Application not installed or not opening.

2) Critical Defect: Assume that there is defect in the software it will not block Test Engineer to do further testing but this defect is going to affect customer business workflow.

Eg: 1) If I want come auto at 10 am but it will come 11 am.

2) If I recharge for 100 rupees but done for 90 rupees.

3) If I ordered 1 product but I get 2 products.

3) Major Defect: Assume that there is a defect in the software But we don't know how it is going to effect on customer business workflow.

Eg: 1) Pin the chat is not working.

2) Starred message.

3) Last seen hide

4) Minor Defect: Assume that there is defect in the software But this defect will not at all effect on customer business workflow.

Eg: 1) spelling mistake

2) Alignment issue

3) Color and font

PRIORITY

- The importance given to the defect is called as priority.

There are 3 types of Priority:

1) P1/ High 2) P2/ Medium 3) P3/ Low

1) P1/ High: If the defect is having priority P1/ High, it will fixed immediately.

2) P2/ Medium: If the defect is having priority P2/ Medium it should fixed within 1 Or 2 build or within a release.

3) P3/ Low: If the defect is having priority P3/ Low It can be fixed in the upcoming release.

There are 4 combinations:

- 1) High Severity & High Priority
- 2) Low Severity & Low Priority
- 3) High Severity & Low Priority
- 4) Low Severity & High Priority

1) High Severity & High Priority

- The defect causes major system failure and must be fixed immediately.

Example 1: Application Crash on Login

Issue: Users cannot log in to the banking application, causing complete inaccessibility.

Impact: Major functionality is blocked for all users.

Fix Urgency: Must be resolved immediately.

Example 2: Payment Gateway Failure

Issue: Customers cannot make online payments in an e-commerce website.

Impact: Direct revenue loss for the company.

Fix Urgency: Needs urgent attention.

2) Low Severity & Low Priority

- The defect is minor and does not affect functionality or business operations.

Example 1: Spelling Mistake in Terms & Conditions Page

Issue: "Privacy Policy" instead of "Privacy Policy" in the footer.

Impact: No impact on functionality, and most users might not even notice.

Fix Urgency: Can be fixed in future releases.

Example 2: Alignment Issue in Admin Dashboard

Issue: A button is slightly misaligned in the admin panel.

Impact: Only visible to internal users, does not affect the user experience.

Fix Urgency: Not urgent, can be fixed later.

3) High Severity & Low Priority

- The defect is serious but occurs in rare scenarios or a less-used feature.

Example 1: System Crash When Exporting a Large Report

Issue: **The application crashes when exporting a report with more than 1 million records.**

Impact: **Critical issue, but it affects only a few users who export large reports.**

Fix Urgency: **Can be fixed in the next update.**

Example 2: Data Loss in a Rare Edge Case

Issue: When editing a saved form in offline mode and syncing, the data is lost.

Impact: Major issue, but only affects users who work offline (which is rare).

Fix Urgency: Needs a fix but is not an immediate priority.

4) Low Severity & High Priority

- The defect is minor but affects an important business area or branding.

Example 1: Company Logo Not Displaying on Home Page

Issue: The company's logo is missing on the homepage.

Impact: Branding issue, users may find the website unprofessional.

Fix Urgency: Needs a quick fix.

Example 2: Typo in a Bank's Interest Rate Advertisement

Issue: "Interest Rate: 10.5%" is displayed instead of "1.05%."

Impact: **Could cause user confusion or legal issues.**

Fix Urgency: **Must be corrected immediately.**

DEFECT

1. Defect

- a. Any feature is not working according to requirement specification is called defect.
- Or
- b. Deviation from requirement specification is called as defect.

Why we get defects?

- c. Wrong implementation.
- d. Missing implementation.
- e. Extra implementation.

Difference between Defect, Bug, Error & Failure?

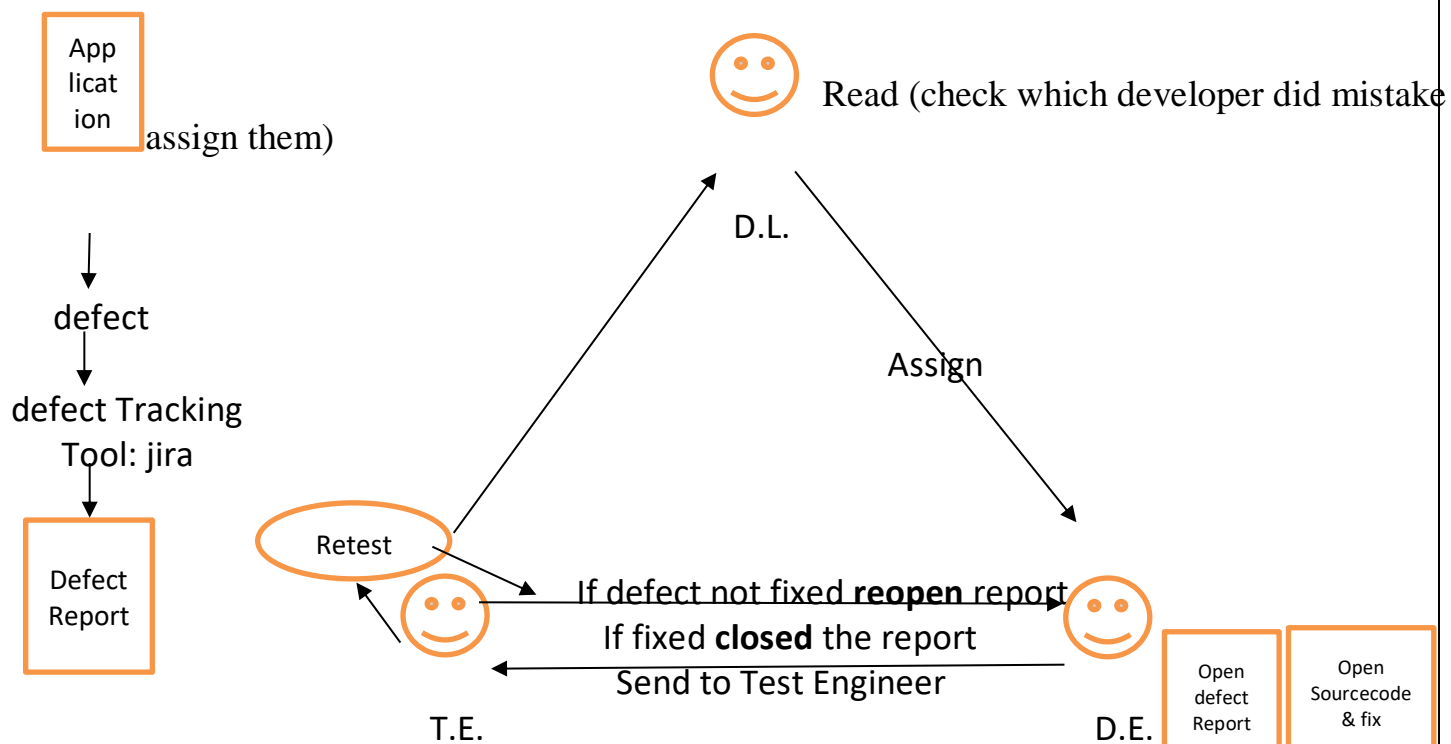
- 1. Defect: Any feature which is not working according to requirement specification.
- 2. Bug: It is an informal name given to the Defect.
- 3. Error: Error is a mistake done by programmer because of which, he will not be able to compile and run the program.
There are 2 types of Error: 1) Compile time Error & 2) Run time Error
- 4. Failure: Defect in the software leads to failure once after it is given to the customer.

2. Defect Tracking Process: Answer in Interview

"The defect tracking process starts with identifying a defect during testing. Once identified, we log it into a defect tracking tool like JIRA or Firelink with detailed information such as severity, priority, steps to reproduce, and screenshots. The defect is then reviewed and assigned to a developer for fixing. After the fix, we perform retesting and regression testing to ensure the issue is resolved and hasn't affected other functionalities. If everything is working fine, we close the defect; otherwise, we reopen it. This process ensures defects are managed systematically and resolved efficiently."

Defect Logging:

- f. Document the defect in a defect tracking tool (e.g., JIRA, Bugzilla, Firelink, etc.).
- g. Provide detailed defect information:
- h. 1)Defect ID 2) Defect Description 3)Severity & Priority 4)Steps to Reproduce 5)Expected vs. Actual Result 6)Screenshots/Logs (if applicable)

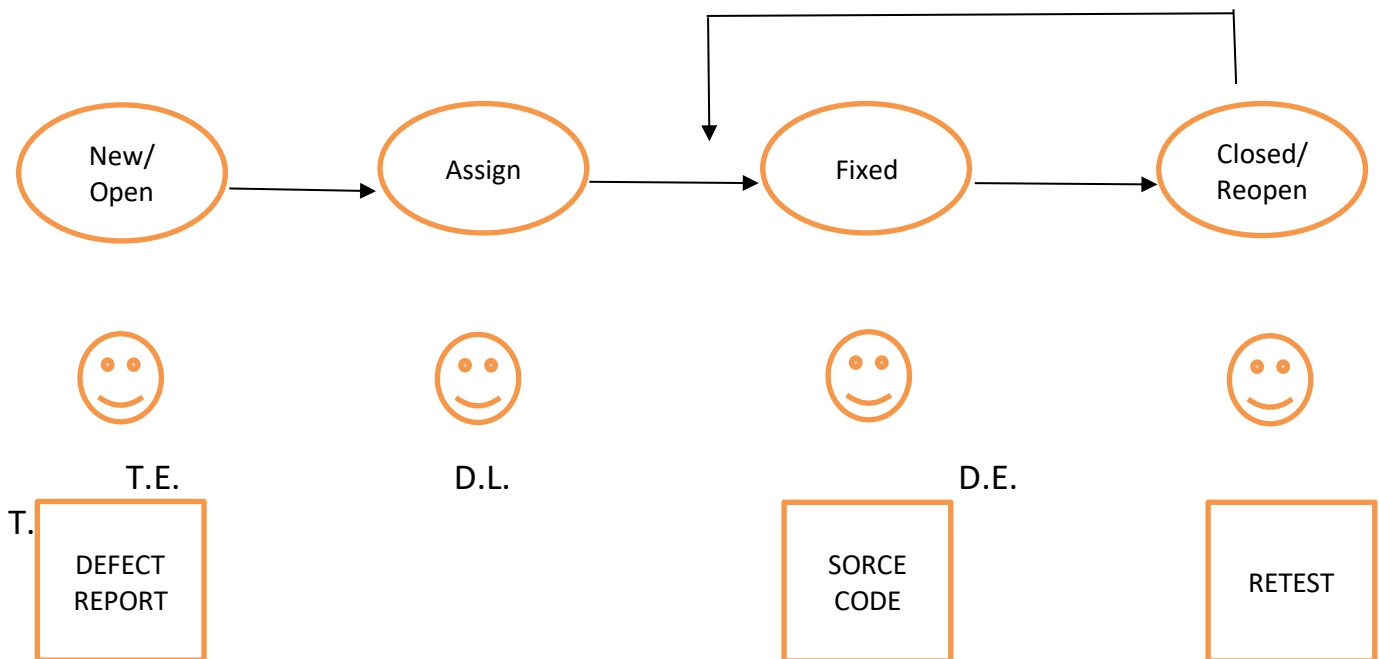


3. DEFECT LIFE CYCLE/ BUG LIFE CYCLE:

- i. The **Defect Life Cycle (Bug Life Cycle)** is the process a defect goes through from identification to closure.
- j. It consists of several stages to ensure proper tracking, fixing, and verification.

Answer in interview:

"The Defect Life Cycle consists of multiple stages, starting from when a defect is found. It is first marked as 'New' and assigned to a developer. The developer analyzes and fixes the defect, updating its status to 'Fixed.' After that, testers retest the defect, and if it's working fine, it moves to 'Closed.' If the defect is not resolved properly, we reopen it. In some cases, defects might be deferred for a future release or marked as duplicate/rejected. This process ensures that defects are tracked efficiently and resolved systematically."



Status in Defect life Cycle:

- 1) New/ Opened 2) Assign 3) Fixed 4) Closed 5) Reopen 6) Reject 7) Duplicate 8) Cannot be fix
- 9) postpone/ Deferred 10) Not Reproducible 11) Request for enhancement

1) New/Open: A tester finds a defect and logs it into a defect tracking tool.

Example: The "Add to Cart" button is not working on an e-commerce website.

2) Assign: The test lead reviews the defect and assigns it to a developer.

Example: The test lead assigns the "Add to Cart" issue to Developer A.

3) Fixed: The developer has fixed the issue and marked it as resolved.

Example: The missing API call for the "Add to Cart" button is added, and the defect is marked as "Fixed."

4) Closed: The defect is successfully fixed, tested, and no longer exists.

Example: The defect is verified and marked as "Closed."

5) Reopen: The defect is not fixed properly or appears again in another scenario.

Example: The "Add to Cart" button works on Chrome but still fails on Safari.

6) Reject: T.E. will find defect and sent to the D. Lead/ Team, they will say it is not a defect it is a feature.

Why we get: 1)Because of misunderstanding the requirement. 2)if the s/w is wrongly installed.

Example: if "username" textfield accepting numbers, and there is requirement is "it should be alphanumeric"

7) Duplicate: If the defect is already send and tracked by someone else at that time we will get duplicate status.

Why we get: 1)Because testing the common feature.

2)Old test engineer has already found some defect and send it to development team in that some defects are fixed & some defects are pending. New test engineer will join the same project and communicate old defects, then we get status duplicate.

How to avoid duplicate status: Before the prepare a report make sure it is not duplicate, You should search for duplicate report with the help of tools by using certain keywords.

8) Cannot be Fixed: T.E. will find the defect and send it to D. Team/ Lead, they will agree yes it is defect but due to some reason(low priority, complex fix) they cannot fix it.

Why we get: 1)If there is minor defect in the root of the product at that time we get cannot be fixed status.

2)The technology itself doesn't support.

3)If cost of fixing the defect is more than cost of defect at that time we get cannot be fixed status.

9) Postpone/ Deferred: T.E. will find the defect and send it to D. Lead/ Team, they will agree it is defect, But they need some time to fix the defect.

Why we get: 1)If it is a Minor defect and developer is not having sufficient time to fix the defect in this case we get Postpone/ Deferred status.

2)If T.E. find the defect in a feature where customer wants to do many changes or remove the feature at that time we get postpone status.

3)If T.E. finds a defect which is expose to internal user and it is a major/ minor defect at that time we get Postpone/ Deferred status.

10) Not reproducible: The defect which is found by the T.E. & cannot seen by developer.

Why we get: 1)Because of incorrect defect report.

i)Not mentioning platform ii)not mentioning correct data

11) Request for Enhancement: T.E. will find the defect and send it to D. Lead/ Team, Developer will say that it is not a defect because it is not mentioned in the requirement.

k. Developers will talk to the customer regarding adding a new feature at that time we get Request for Enhancement status.

INTERVIEW QUESTIONS ON DEFECT:

- 1) What is defect ? and give example for defect?
- 2) what is difference between Error, defect, bug, failure
- 3) what is severity? Explain types of severity with example.
- 4) what is priority and explain types of priority ?

5) Give examples for below combinations:

example High Severity and Low Priority example High Severity and High Priority example Low Severity and High Priority example Low Severity and Low Priority

6) who will give severity and priority?

answer: TE will give severity and priority. But in some companies Developers, BA and PM can change the Priority.

7) Defect leakage: defect missed by TE and found by the customer is called as defect leakage.

8) defect masking/Fault Masking: one defect hides the other defect.

9) defect release: Releasing software to the customer with the set of known defects is called as defect release. These defects will be having low severity and low priority

10) Defect Seeding: The process of introducing the defect intentionally into the software is called as defect seeding.

11) defect cascading: one defect triggers the other defect is called as defect cascading/ defect caused by the other defect.

12) defect clustering: some of the modules contains most of the defects during testing and these defects will not be entire software failure. To overcome these TE should identify impacted modules and find the defects in the early stage

13) what is 'Latent Defect'

Answer: defect present in the software and not identified for particular period of time

Fatal defects :(Blocker/Showstopper)

It is the other name given to the blocker defect because of this defect test engineer will be completely blocked to test or showstopper or DEFECT defect.

14) Deferred defect :(minor defect/trivial defect)

It is the other name of minor defect which will NOT effect the customer business workflow and this defect is

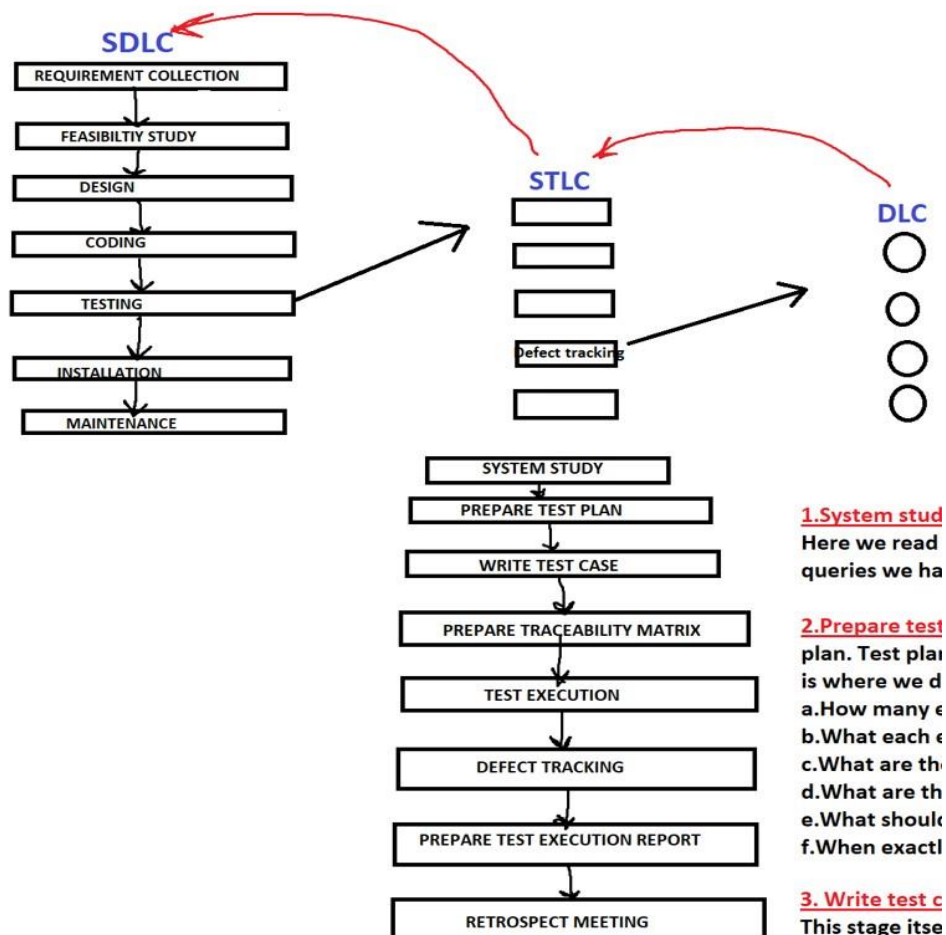
15) Defect leakage / Bug leakage :

Defect missed by the Test Engineer but caught by customer is called as Defect Leakage or Bug leakage.

16) Bug release / defect release : Releasing the software with known set of defect is called as Bug release or defect release. Here we should make sure defect are minor and meets the acceptable limit.

SOFTWARE TEST LIFE CYCLE (STLC)

- STLC is a procedure to test the software.
- STLC is a part of SDLC
- Defect life cycle is a part of STLC



1. System Study:

Read the requirement, understand the requirement if you have any queries, interact with BA, developers or customer.

2. Prepare Test plan:

Once after reading and understanding the requirement, we go for preparing the test plan.

Test plan is a document which drives all the future testing activities:

- a. Here we decide how many engineers we require to complete the testing.
- b. What is the total time for completing the testing/project.
- c. What each engineer should do in different stages of testing.
- d. What are types of testing we will conduct in future.

- e. What are the features that are to be tested and not to be tested
- f. What is the testing approach
- g. When each activity should start & end.

3. Write test cases:

Test case is a document which contains all possible scenarios. This stage itself has got different activities like:

- i. Identify all possible scenarios
- ii. Write test case
- iii. Review test case
- iv. Fix the review comments
- v. Verify the fix
- vi. Test case approval
- vii. Store the test case in test case repository

4. Prepare traceability matrix:

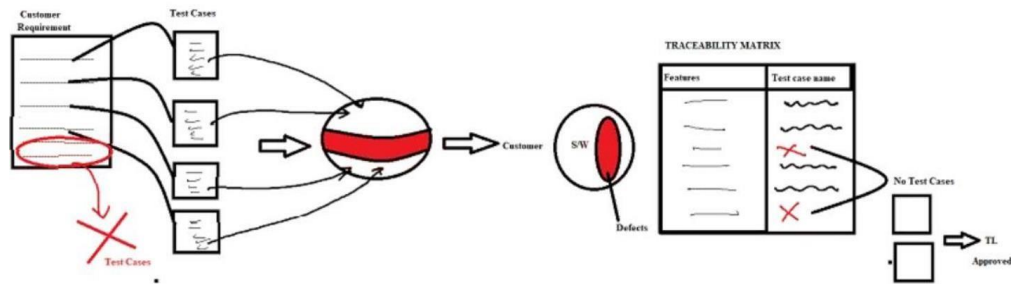
Once after we write test case, the biggest question is what is the proof that each and every requirement has got a test case?

We prepare traceability matrix to ensure that each and every requirement has got at least one test case.

Traceability Matrix/ Requirement Traceability Matrix (RTM)/ Cross Reference Matrix (CRM) What is

Traceability Matrix?

It is the document which we prepare to make sure that every requirement has at least one test case.



Sl	No	Module Name	High level requirement	Detailed Requirement	Test Case Name	Automation
1.		Loan	1.1 Personal Loan	1.1.1 ----- 1.1.2 -----	CB_PL_Approval CB_PL_Rejection	CB_PL_Approval CB_PL_Rejection
			1.2 Home Loan	1.2.1 ----- 1.2.2 -----	CB_HL_Approval CB_HL_Rejection	No No
2.		Amount Transfer	2.1 FAN text field	2.1.1 should accept only 10 digit numbers 2.1.2 should accept only those numbers which is created by manager.		No No
			2.2 TAN text field	2.2.1 ----- 2.2.2 -----		Yes Yes
			2.3 Amount text field	2.3.1 should accept only positive integer b/w 100-5000		

Advantages:

1. It ensures that every requirement has got at least one test case, which indirectly assures you that you are testing every feature at least once
2. It gives traceability from high level requirement till automation script name

Drawback

1. It will not ensure that you that got 100% coverage

Types of Traceability Matrix:

There are 3 types of traceability Matrix.

1. **Forward Traceability Matrix:** Mapping from the root document to derived document is called forward traceability matrix.
Ex: Mapping from Req to test case and test case to automation script
2. **Backward Traceability Matrix:** Mapping from derived document to root document is called as Backward Traceability Matrix.
Ex: Mapping from test scripts to test cases and test cases to requirement.
3. **Bi-Directional traceability Matrix:** Doing both forward and backward traceability matrix is called as Bi-directional traceability Matrix

Interview Questions:

Q.1 Difference between Traceability Matrix and Test case review:

<i>Traceability Matrix</i>	<i>Test case review</i>
Here we check every req has got at least one test case.	Here we check test case is covering all possible scenarios for specific requirement.
Here we don't check whether test case is covering all possible scenarios for a specific requirement.	Here we don't check whether every requirement has got at least one test case.

Q.2 If there is no requirement then how will you prepare traceability matrix?

I will explore the application and understand it, identify the scenarios, document it, write test cases. Once after writing the test cases what is the guarantee that you have written all the test case for all the features, to ensure that we write traceability matrix Here we list all the features that are there in the product and map it to test case.

<u>Features</u>	<u>Test case name</u>
Login	-----
Signup	-----
Forgot password	No

Q.3 Assume that last day when you are about to release the product to the customer, if there is a critical / blocker defect what will you do? Will you release product to the customer or not?

As a TE I'm not a decision maker so I will list all the defects which are pending and the impact of the defect on the business also I suggest whether to release or not and send it to PM. It is the manager who should take a calculated decision and decide

5. Test Execution: This is the stage wherein we test the product for 4-8 cycles (if it is agile) and 30-60 cycles (if it is traditional model).

- This is the stage where we execute all the test cases.
- This is where we conduct all types of testing
- This is where we catch the bugs and help the developers to improve the quality of the product. This is where TE are productive to the organization

- This is the stage where the T.E spends a lot of time
- This is the stage where test engineers become productive to the organization

6. Defect tracking:

Once after test execution, obviously we are going to find the defects. Each defect that we find should be tracked in an organized way. This is called as Defect tracking.

7. Test execution report:

At the end of every test cycle we prepare test execution report. It is a document which we prepare and provide to the customer at the end of every test cycle.

This report covers:

- Total no. of test cases
- Total no. of test cases executed.
- Total no. of test cases not executed. □ No of test cases passed. □ No of test cases failed □ What is the pass percentage?
- What is the fail percentage?

We will prepare this document and send it to the customer. From customer's POV this is the end of the project. But from company's POV we have one more activity called as Retrospective meeting.

8. Retrospective meeting/Project closure meeting/Post-mortem meeting:

Here the entire team will sit together and discuss about the list of achievements and mistakes, they document all this, that document is called as Retrospect document. In the next release/sprint they open this document in the planning stage and plan in such a way that all the achievements are adopted and all the mistakes are avoided.