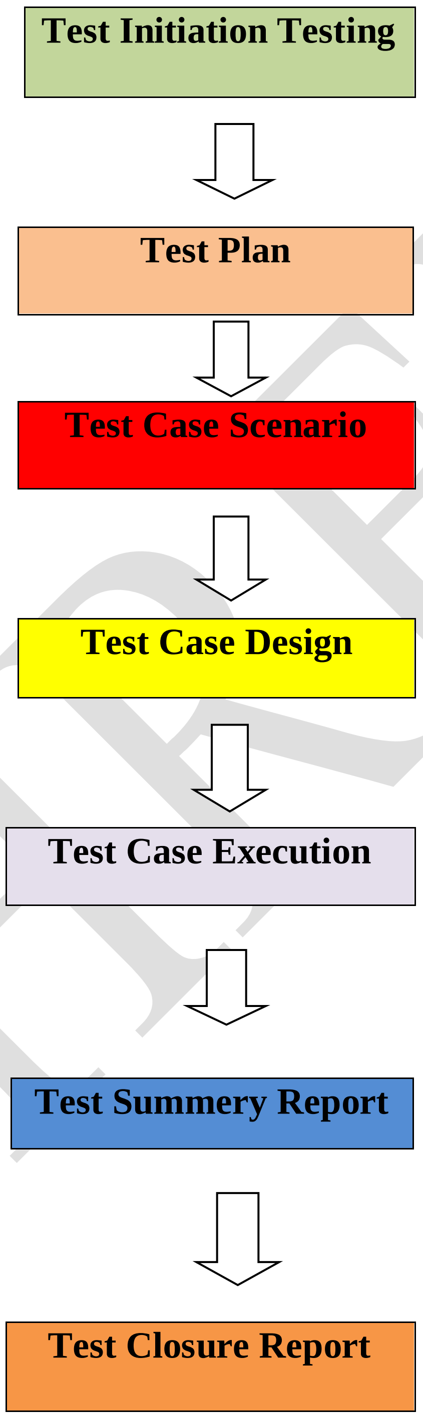
Manual Testing Part-2

Software Testing Life Cycle (STLC):



STLC Stages:

1) Test Initiation Testing:

A) Requirement of the Project:

-Requirement of project means domains of the project
1) Banking Domain 2) Telecom Domain 3) E-Commerce 4) E-Educational 5) Health Care 6) Insurance Domain

B) Scope of the Project:

-Strategy and Methodology of the project will be decided
-Methodology means which type of testing methodology Automation testing or Manual Testing
-Project owner/Product Owner involve here

C) Risk Involved in the Project

-Less Resources: if less no. of resources involve then person need to do extra work/efforts
-Less Test Data: If there less data or no test data then we perform "Adhoc Testing"
-Lack of knowledge: KT (Knowledge Transfer) will be provided

Test Plan

A) Resources Allocation

-As per scope of the project test methodology selected and then as per that
-Work will allocate to specific tester (work of Test Team Leader**)**

B) Estimation of Time (Start date and End date of the project will be decided

Test Scenario:

-Means **“What to test”**

-Test Means **"Validate/Verify"** and Scenario Means **"User Journey".**

-It is also called **"Test Condition"** or **“Test Possibility”**

4) Test Case Design:

-Test Cases are **“how to be tested”.**

-Test Case Design means Steps involve while testing

-For this we refer either Sprint Backlog Document or Description and Acceptance Criteria Mentioned in Development Ticket

-In that Positive and negative test case are there. (But we just write down positive test cases but while testing we test positive and negative both scenario)

Test Case Execution:

-Once developer local code merged in Master repository of the company then ticket ready for testing.

-Means Application came in **SIT Environment from DIT Environment**.

-Once **developer assigns ticket to testing** with application URL then **we start "Test Case Execution".**

-Here we start testing **as per Test case design steps.**

-If we **found any defect** then **we perform retesting and then raise the bug ticket**.

-Once developer resolved issue then again perform **"Retesting"** and **"Regression Testing".**

Test Summary Report/Test Documentation

-We "Tester" responsible for "Test Summery Report".

-Total how many test case designs

-How many test cases **execute**

-How many test cases **pass**

-How many test cases **fail**

-We will **create bug ticket** for **failed test cases**.

-We create this report in excel format. Once this Test report created then we send to Testing Team Leader.

For example:-

	Total Test Case Design	**Test Cases Executed**	**Test Cases Passed**	**Test Cases Failed**
Daily	**15**	**15**	**14**	**1**
Monthly	**300**	**280**	**265**	**15**

Test Closure Report

-"Team Leader" is responsible for to make **Test Closure Report**

-In this closure report he checks whether all process are correct or not.

Activities: 1) Analysis of Test Summary Report 2) Analysis of Bug Ticket/Report

-Once Test Closure Report created then **Team leader** send it to **Test Manager/Product Owner/Project owner**.

What is Test Case Review?

-Definition: Recheck test cases after writing.

Types of Review:

1) Self Review:
-Review going to done by our self after creating/designing of test cases.

2) Peer Review:
-Review going to be taken by your Colleague/team member/Senior Member.

3) Internal Review:
- Involvement: [Tester + PO (Product Owner)] +Developer

4) External Review:
-Perform by the client during UAT.

-Involvement: Development Team + Testing Team + PO + Client

-Also called Walkthrough/Inspection

How you are going to know about good test cases?

1) Should be simple, easily understandable.
2) Complete all Functionalities (Means here no functionality miss from Testing Team)
3) Avoid test case repetition
4) Do not assume Functionality
5) Create test cases with end user in mind.
6) Test case must be identifiable (Test Case ID)

Requirement Traceability Matrix (RTM)

-It is also known as Traceability Matrix

-It is mapping between prepared test cases and business requirements.

-RTM Document is generally used to ensure the test Coverage.

-If all requirements are mapped with the test cases then we can ensure test coverage.

-This document prepare before the test case execution.

-Due to this testing team will not miss any functionality during testing phase from the requirement specification.

There are mainly two types of RTM:

Forward Traceability Matrix:

-Actually it is mapping between prepared test cases and business requirements.

Backward Traceability Matrix:

-Actually it is mapping between Defect and business requirements.

What is a Test Case?

-Test Cases are **"How to be Tested" functionality?**

Test case – is the smallest unit of the testing plan – which includes a description of necessary actions and parameters to achieve and verify the expected behaviour of a particular function or the part of the tested software.
So you need to understand where to start testing, which general steps need to be executed and what the result should be. And then this scenario is broken down into more detailed parts – test cases – to define all positive, negative, localisation and other behaviours of the software

Example:

testers need to test the functionality of uploading photos.

We create first test scenario as:
1. The user must be logged
2. Move to the "upload photos" page
3. Click the "upload" button
4. Select photos
5. Upload them

Now, **this scenario should be divided into detailed test cases**,
Example:

•Check the logged user possibility to go to the "upload photos" page
•Check the not logged user possibility to go to the "upload photos" page
•Check whether the user can click "upload" button
•Is it opens a form to select a photo and possibility to close it
•What happens if you do not select photos, choose another file format (for example video), choose photos of a

maximum size and so on
•Check the possibility to upload photos
•Check if photo is saved
•Possibility to reload or delete photos
•What happens with photos in the case of the disappearance of the Internet or the device is switched off
•Are all buttons displayed correctly at another location or on different operating systems (if any difference)
And so on…
-The number of test cases depends on the experience and imagination of the tester.

-Therefore, the process of writing test cases starts from forming a test scenario or user story, and then it can be divided to check different occasions.

Test Case Design Technics

Static Test case Design Technic

(Related to Quality Assurance Developer involved)

Dynamic Test Case Design Technic

(Related to Quality Control Tester Involved) -
Also called Black Box Testing Technic.
Types:
A) Boundary Value Analysis
B) Equivalence Class Portioning
C) Decision Table
D) Cause And Effect Graph
E) State Transition Testing

Note: from Above technics in our company we just follow "Boundary Values Analysis" and "Equivalence Class Portioning".

A) Boundary Value Analysis (BVA)
-Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.
-It is generally seen that a large number of errors occur at the boundaries of the defined input values rather than the centre. It is also known as BVA and gives a selection of test cases which exercise bounding values.
-This black box testing technique complements equivalence partitioning. This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

Guidelines for Boundary Value analysis
1) If an input condition is restricted between values x and y, then the test cases should be designed with values x and y as well as values which are above and below x and y.
2) If an input condition is a large number of values, the test case should be developed which need to exercise the minimum and maximum numbers. Here, values above and below the minimum and maximum values are also tested.
-Means here we check min, min-1, Min+1, max, max-1, max+1
Example: Password field accepts minimum 8 characters and maximum 12 characters

Allow 1 Capital Char, 1 Small Char, special Symbol & Number, Space not allowed
So here BVA Size is a
Min=8, Max=12, Min+1, Max-1 will be **Accepted**
Min-1, Max+1 will not be Accepted **(Rejected)**

B) ECP (Equivalence Class Partition)

Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same. This software testing method divides the input domain of a program into classes of data from which test cases should be designed.
The concept behind this technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to identify valid as well as invalid equivalence classes.
Example: Password field accepts minimum 8 characters and maximum 12 characters
Allow 1 Capital Char, 1 Small Char, special Symbol & Number, Space not allowed
So here Equivalence Partition is
Valid ECP: 0-9 Number, a-z Char, A-Z Chart, Special Symbols
Invalid ECP: (-) Space not allowed

Test Case Design Structure:

Step	Test Case Field	Description
1.	Test Case ID	Each test case should have a unique ID.
2.	Test Priority	It is useful while executing the test. 1)Low 2) Medium 3) High
3.	Test Designed By	Testers Name
4.	Date of test designed	Date when test was designed
5.	Test Executed by	Who executed the test(tester)
6.	Date of the Test Execution	Date when test needs to be executed
7.	Name or Test Title	1. The title is important because it's often the first or only thing you see when you are scanning a list of test case 2. Clear titles are the key to help testers to find quickly the right test cases.
8.	Name or Test Title	A detailed description of the test case. In this section, you can also set up categories to organize your test cases into logical groups.
9.	Pre-condition	Any requirement that needs to be done before execution of this test case
10.	Test Steps	1. Test Steps section gives the tester a numbered list of the steps to perform in the system, which makes it easier to understand the test case.

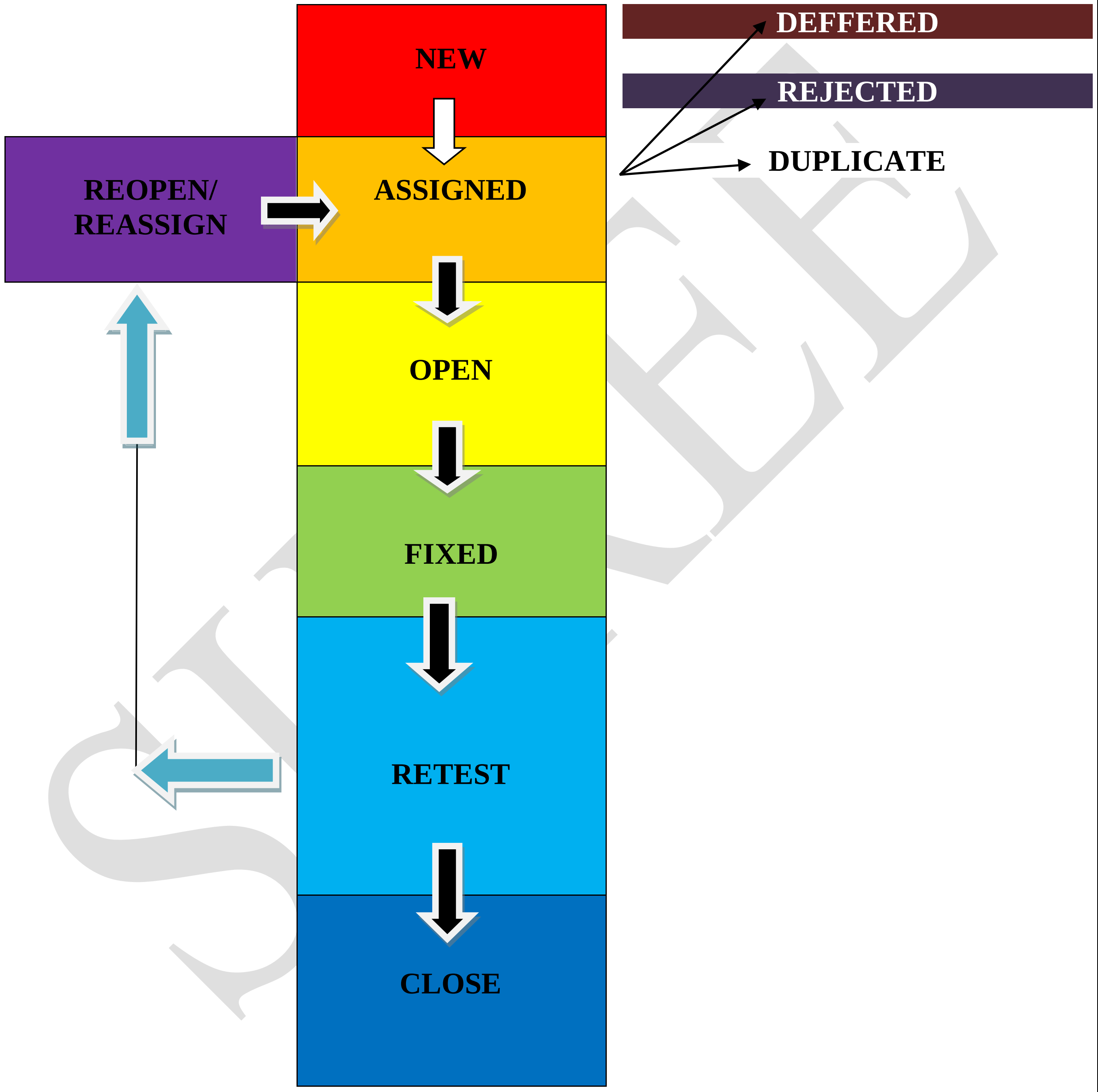
		2. 8 test steps per one test case. Too many steps make it difficult for developers and testers to reproduce the steps when a bug report is filed against the test case
11.	Test Data	You can enter test data directly in the test data field, or refer to a separate file that contains test data for one or more test cases.
12.	Expected Results	1. Mention the expected result including error or message that should appear on the screen. 2. The tester needs to know the expected result in order to assess whether the test case is successful**.**
13.	Post-Condition	What would be the state of the system after running the test case
14.	Status (Fail/Pass)	Mark this field as failed, if actual result is not the same as the expected result
15.	Notes/Comments/Questions:	If there are some special conditions which is left in the above field
16.	Requirements	List of the requirements for a particular test cycle
17.	Attachments/References	The files and documents that are attached to the test case, such as screen captures and other supporting material
18.	Automation? (Yes/No)	Fill “YES” when test cases are automated

Note:

-Above structure is standard structure of test case design but Organisation wise somehow it varies. In above structure given lot of test case field but not all field used in Organisation.

-Below we have given 2 test case design template of our companies for your reference.

Defect/Bug Life Cycle:



Bug Life Cycle/Defect Life Cycle

New: When we create bug ticket first time then that time Status is "New"

Assigned: Then we assign issue to specific Developer then status will be "Assigned"

Open: When Developer doing Coding changes for fixing the issue

Fixed: When Developer fixed or resolved issue through Code changes

Retest: Once it fixed then again assigned to us for Retesting

Closed: We verify issue if issue not present then we closed that ticket.

Reopen: If issue not resolved after developer work then we again re-assign to Developer

Note: So 1-7 Point is Regular Bug life Cycle so in interview just explain this. If Interviewer asked specific about below Status point then explain below points.
There are also some Bug ticket Status

Deferred:
Means it is known issue.
-But this issue is Low Priority and Severity.
-It's not make much effect on functionality of application (Means not Stopper or Blocker issue)
-Then PO will decide this issue will fix in next Release or Sprint.

Rejected:
If we raised bug and Developer not accept is as a Bug then he/she will reject this bug.
-Then that we can discuss with PO at the time Scrum Meeting with All details information with Screenshot/video of defect.
-Then PO check Customer Requirement then took decision accordingly.

Duplicate:
means if we found same issue twice and created ticket for that then that ticket will be "Duplicate"
-In that condition one ticket will be cancelled as Duplicate ticket