**SOFTWARE TESTING**

**DOCUMENTATION**

**TABLE OF CONTENTS**

**1. Introduction**

**2. SDLC**

**3.Verification vs Validation**

**Client side validation**

**server side validation**

**4.STLC**

**5.Testing levels**

**Functional testing**

**Unit Testing**

**Integration Testing**

**Smoke Testing**

**Sanity Testing**

**System Testing**

**Regression Testing**

**User Acceptance Testing – UAT**

**Alpha Testing & Beta Testing**

**Non-Functional testing**

**Performance Testing**

**Load Testing**

**Stress Testing**

**Security Testing**

**6.Testing Methods**

**Black box**

**white box and**

**Grey box testing**

**7.Positive and Negative testing**

**BVA and EP**

**8.Database testing**

**9.GUI testing**

**10.Agile Testing**

**11.API Testing**

**12.Mobile Testing**

**13.Severity and Priority**

**14.Test metrics**

**15.Test Plan**

**16.Test case template and preparation**

**17.Test management tool**

**18.Requirements management tool**

**19.Defect tracking tool**

**20.Bug life cycle**

**21.Information while creating a bug**

**1.Introduction**

**Software Testing :**

Software Testing is the process of identifying the correctness and quality of software program. When a bug or defect causes in software application, testing is done to find out the cause of defect and to remove the bug.

**Software Testing ways:**

Manual Testing

Automation Testing

**Manual Testing:**

Manual Testing is a process of finding out the defects or bugs in a software program. New applications must be tested manually before getting automated. As manual testing involves complete test cases it is a time consuming test.

**Automation Testing:**

Automation Testing means using an automation tool to execute your test case suite.The goal of Automation is to reduce the number of test cases to be run manually and not to eliminate Manual Testing altogether.

**When to automate?**

High-risk.

Test cases that are repeatedly executed

Difficult to perform manually

Time-consuming.

**Advantages:**

Fast

Repeatable

Reusable

2.**SDLC**

SDLC is Software Development Life Cycle. It is the sequence of activities carried out by Developers to design and develop high-quality software.

It does not involve just coding tasks done by developers but also incorporates the tasks contributed by testers and stakeholders.

**Life cycle of software development:**

Project Initiation

Requirements

Design

Implementation

Testing

Deployment

Maintenance

**Project Initiation:**

It involves starting up a new project. You can start a new project by defining its objectives, scope, purpose and deliverables to be produced. It's the phase in which you define your scope and hire your team.

The senior IT Stakeholder and senior Customer Stakeholders must be identified for the purpose of making high level decisions throughout the SDLC. They must work with the customer to ensure a project sponsor is in place to support the effort.

**Requirements:**

Gather as much information as possible about the details & specifications of the desired software from the client.

**Design:**

In this phase, program developer analyses whether software can be prepared to fulfill all the requirements of the end user. After that best design approach is selected for the product. The developer selects the program language like Java, Oracle etc. which will be best suited for the software.

**Implementation:**

Development team does the actual coding based on designed software and writes unit tests for each component to test the new codes written by them. It is the largest phase in SDLC.

**Testing:**

This is the last phase of SDLC before the software is delivered to the customer. The job of test team is to test the system against the requirements. The aim of tester is to find out the gaps or defects within the system and also to verify that the software works as expected according to the requirements.

**Deployment:**

Once the Product is tested and ready to deploy, it is released to consumers to use.

**Maintenance:**

The process where the care is taken for the developed product is known as Maintenance. You might require to change the code as per customers requirements.

**Models of Software Development cycle:**

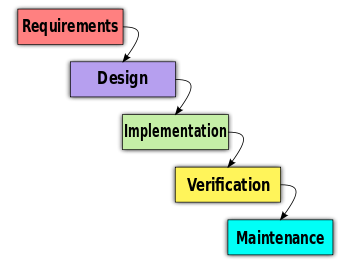
WaterFall Model

Iterative Model

V-Model

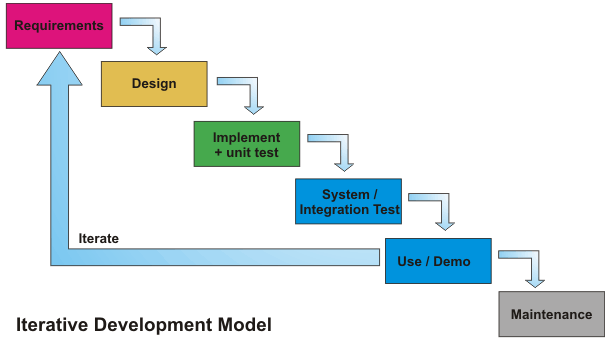
**WaterFall Model:**

Waterfall model is a sequential model divided into different phases of software development activity  flowing steadily downwards. Each stage is designed for performing specific activity during SDLC phase.

****

**Iterative Model:**

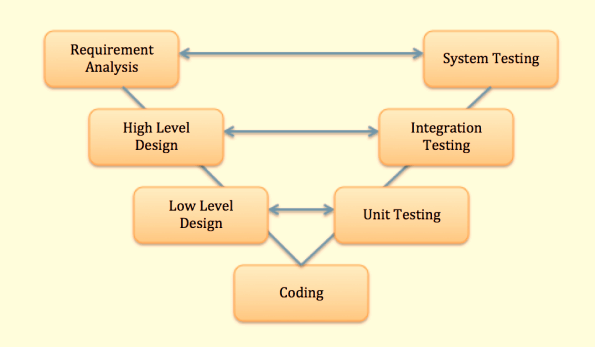
Iterative model is which breaks down larger applications to smaller chunks. The feature code is designed, developed and tested in separate cycles.



**V-Model:**

In V-Model execution of processes happens in a sequential manner in V-shape. It is also known as Verification and Validation Model.

It has an association of a testing phase for each corresponding development stage. In the V-Model, each stage of verification phase has a corresponding stage in the validation phase.



**Typical Phases Of Validation In The V-model:**

Unit testing

Integration testing

System testing

**3.Verification and Validation**

**Validation:**

It is concerned with checking that the system will meet customer’s actual need.

Client-side validation

Server-side validation

|  |  |
| --- | --- |
| Client-side validation | Server-side validation |
| Validation done using javascript | Validation is done at the server side where application resides |
| Fast and easier for use | Slow for user |
| Insecure as the end user has access to the code of the page | Secure |
| Pop-up is shown | No pop-up |

**Verification :**

It checks whether the system is error-free.

**4.STLC**

STLC is Software Testing Life Cycle. Software Testing Life Cycle is defined as a sequence of activities conducted to perform Software Testing.

**Lifecycle of Software Testing:**

Requirement Analysis

Test Planning

Test case Development

Environment Setup

Test Execution

Test Cycle Closure

**Requirement Analysis:**

During this phase, test team studies the requirements from a testing point of view to identify the requirements.

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do)  or Non Functional (defining system performance /security availability )

**Test Planning:**

Senior QA lead will determine effort and cost estimates for the project and would prepare and finalize the Test Plan.

**Test Case Development:**

Here test cases and test scripts are created, verified and reworked.

**Environmental Setup:**

Test environment decides software and hardware conditions under which a product is tested.

There are 3 Environment types:

QA(Quality Assurance)

Stage

Product

In QA and stage we can edit requirements.

**Test Execution:**

Bugs will be reported back to the development team for correction and retesting will be performed.

**Test Cycle closure:**

Applications to be implemented in future.

**5.Testing Levels**

Any type of software testing type can be executed both manually as well using an automation tool.

Functional Testing

Non-Functional Testing

**Functional Testing:**

Unit Testing

Integration Testing

Smoke Testing

Sanity Testing

System Testing

Regression Testing

User Acceptance Testing – UAT

Alpha Testing & Beta Testing

**Unit Testing:**

Unit Testing is done during the development of an application.

The goal of Unit Testing is to isolate each part of the program and show that the individual parts are correct. Used by developers

**Integration Testing:**

Individual software modules are combined together and tested as a group.

**Smoke Testing:**

To ascertain that the critical functionalities of a program is working fine.

Performed after software build

To reject a badly broken application

**Sanity Testing:**

Sanity testing is done to check the new functionality/bugs have been fixed.

**System Testing:**

System Testing is the testing of a complete and fully integrated software product.

**Regression Testing:**

Changes made to few parts of the code does not change any functionalities on the system.

**User Acceptance Testing – UAT:**

It is performed by the client to certify the system with respect to the requirements.

Final phase of testing before moving the software to the market.

Is also called End-User and Operational Acceptance Testing.

**Types:**

Alpha Testing

Beta Testing

***Alpha Testing***: Done Onsite- Developers as well as business analysts are involved with testing team.

***Beta Testing***: Done client Side- By real users or customers. Developers and business analysts are not included.

**Non-Functional Testing:**

Non-Functional Testing is also known as **Performance Testing.**

It is to verify the stability of the system under test.

It is to establish a benchmark behaviour of the system.

**Techniques:**

Load Testing

Stress Testing

Security Testing

Reliability Testing

***Load Testing:***  It is meant to test the system by constantly and steadily increasing the load on the system till the time it reaches the limit to which the system can handle.

The primary goal of load testing is to define the maximum amount of work a system can handle.

***Stress Testing:***The purpose behind stress testing is to find out  the failure of system and to monitor how the system recovers back gracefully.

***Security Testing:*** It is done to check whether the application or the product is secured or not.

***Reliability Testing:*** It is the failure which are discovered and removed before deployment.

**6.Testing Methods**

**Black box**

**white box and**

**Grey box testing**

***Black box Testing:***

Black Box Testing method testing is done without knowing the internal codes and structure of the program.

It is done from customer’s point of view and the tester knows only about the inputs and the expected outputs of the application.

It is also known as **Functional Testing**.

**Black Box Testing Techniques:**

There are different techniques involved in Black Box testing.

Equivalence Class

Boundary Value Analysis

**White box Testing:**

It is the testing method in which internal codes & structure of the software is known to the tester.

**White Box Testing Techniques:**

***Statement Coverage*** – It is aimed at exercising all programming statements with minimal tests.

***Branch Coverage*** – Running a series of tests to ensure that all branches are tested at least once.

***Path Coverage*** – Testing all possible paths which means that each statement and branch is covered.

**Grey box Testing:**

It is the combination of white box testing and black box testing.In the **Gray Box testing** the tester has knowledge of some parts of internal structure.

**7.Positive and Negative Testing**

**Positive Testing:**

Can be provided on the system by providing valid data as input.

**Negative Testing:**

Can be provided on the system by providing invalid data as input.

**Techniques for positive and negative validation:**

Boundary Value Analysis

Equivalence Partitioning

***Boundary Value Analysis:***

Includes values at the boundary.

Inside limits= positive testing.

Outside limits=Negative Testing.

***Equivalence Partitioning:***

Divides inputs into many partitions.

Valid values =Positive testing

Invalid Values= Negative Testing

**8.Database Testing**

Checking schema, tables,triggers,etc., of the database under test.

Performs back-end testing.

**9.GUI Testing**

Graphical User Interface

Interaction between user and application

**10.Agile Testing**

Agile method proposes **incremental and iterative approach** to software design.

The agile process is broken into individual models that designers work on.

Error can be fixed in the middle of the project.

Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less.

Testers and developers work together.

It requires close communication with developers and together analyze requirements and planning.

**Agile process flow:**

**Concept-** Projects are envisioned and prioritized.

**Inception-** Team members identified, finding initial environments and requirements.

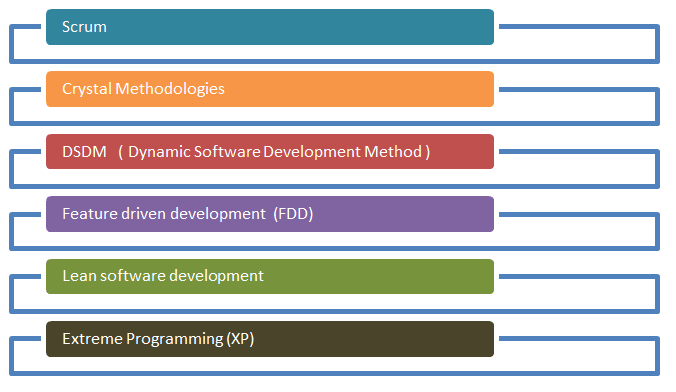
**Iteration-** Works to deliver software based on iteration request and feedback.

**Release-** QA testing, document development and final release into products.

**Production-** Ongoing support of software.

**Retirement-** End of activities.

**Agile Testing Methods:**

****

**Scrum:**

Scrum is a part of Agile movement. It is a lightweight process framework for agile development, and the most widely-used one.

Scrum is most often used to manage complex software and product development, using iterative and incremental practices.

**Sprints:**

Scrum is divided into sprints. The heart of Scrum is a Sprint

A new Sprint starts immediately after the conclusion of the previous Sprint.

In **Sprint planning**, the work to be performed in the Sprint is planned collaboratively by the Scrum Team.

The **Daily Scrum Meeting** is a 15-minute time-boxed event for the Scrum Team to synchronize the activities and create a plan for that day.

A **Sprint Review** is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.

The **Sprint Retrospective** occurs after the Sprint Review and prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself and create a plan for improvements to be enacted during the subsequent Sprint.

The following **artifacts** are defined in **Scrum Process Framework** -

Product Backlog

Sprint Backlog

Burn-Down Chart

Sprint Chart

***Product Backlog***-Functionalities still remaining to be implemented in future

***Sprint Backlog***- maintain sprints that are to be implemented

***Release Burndown Chart***- maintain releases that are to be done

**11.API Testing(Application Programming Interface)**

The API Testing is Communication and data exchange between two software systems.

Contains functions/subroutines which can be executed by another software system.

It cannot access the source code. All functional issues are tested.If API is not tested properly, it may cause problems not only in the API application but also in the calling application.

Since API lacks GUI ,Testing is performed at message(business) layer

**4 methods involve in Api Testing:**

Get

POST

Delete and

PUT.

**GET**- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data.

**POST**- A POST request is used to create new entity. It can also be used to send data to the server.

**PUT**- Create a new entity or update an existing one.

**DELETE**- Removes all current representations of the target resource given by a URI.

**Types of API’s:**

Most common types of APIs are **Web APIs**. Otherwise known as **Web Services.**

web applications or applications that need to connect to each other via the Internet to communicate.

**Public APIs**-Used to do everything from checking traffic and weather, to updating your social media status, or even to make payments.

**Private Web APIs**-Used by companies to extend their services and capabilities across a broad range of use cases.

Simple Object Access Protocol (SOAP)

Remote Procedure Call (RPC),

Representational State Transfer (REST)

**12.Mobile Testing**

To have a successful mobile app, we need to understand that developing a nice mobile app is not the only requirement. A mobile app needs to be tested thoroughly in order to get the likability of users.

You will mainly come across three types of mobile applications:

Mobile Web

Native App

Hybrid App.

**Mobile Web:** Web apps are not real applications; they are actually websites that open in your smartphone with the help of a web browser.

**Native app:** A native app is developed specifically for one platform. It can be installed through an application store.

**Hybrid app:** Hybrid Apps are a way to expose content from existing websites in App format. They can be well described as a mixture of Web App and Native App.

**13.Severity and Priority**

**Severity:**

Severity is defined as degree of impact a defect has on the development or operation of a component application being tested.

Quality Assurance usually determines the severity level of defect.

By Functionality

Based on technical aspect of the product

Project Manager can change priority

Customer may change it via project Manager

Developers and testers cannot change it

**Critical**

**Major**

**Medium**

**Low**

**Priority:**

Priority is defined as the order in which a defect should be fixed.

Developer should resolve a defect.

Tester may change it later if new facts are uncovered.

No one can change it( Including developers and project Managers)

**Low**

**Medium**

**High**

**14.Test metrics**

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

Improves the efficiency and effectiveness of a software testing process.

**Types of Metrics**

**Process Metrics:** It can be used to improve the process efficiency of the SDLC ( Software Development Life Cycle)

**Product Metrics:** It deals with the quality of the software product

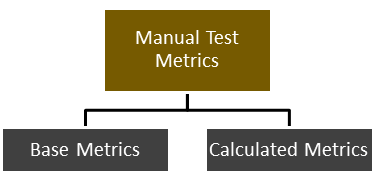
**Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

## **Manual Test Metrics**

Manual test metrics is classified into two classes

**Base Metrics**

**Calculated Metrics**



***Base Metrics:***

Base metrics is the raw data collected by Test Analyst during the test case development and execution

***Calculated Metrics:***

Calculated metrics is derived from the data collected in base metrics. Calculated metrics is usually followed by the test manager for test reporting purpose

**15.Test Plan**

Test plan is to gather all necessary information to plan and control the test efforts related to the project.

Test Manager will start preparing the test plan.

**16.Test case template and preparation**

***Test case Preparation:***

*Test cases can be written directly in JIRA to easily understand the steps for a ticket. It will be helpful for the developers too. Instead of going to test case tool they can see the steps directly in JIIRA. Easy to access in JIIRA rather than in test case tool.*

**Test case ID:**

Unique ID is required for each test case. Follow some convention to indicate types of the test. E.g. ‘TC\_UI\_1’ indicating ‘user interface test case #1’.

**Test priority (Low/Medium/High):**

This is very useful while test execution. Test priority for business rules and functional test cases can be medium or higher whereas minor user interface cases can be of low priority. Test priority should always be set by the reviewer.

**Module Name:** Mention the name of the main module or the sub-module.

**Test Designed By** Name of the Tester

**Test Designed Date:** Date when it was written

**Test Executed By** Name of the Tester who executed this test. To be filled only after test execution.

**Test Execution Date**: Date when the test was executed.

**Test Title/Name:** Test case title. E.g. verify login page with a valid username and password.

**Test Summary/Description:** Describe test objective in brief.

**Pre-condition:** Any prerequisite that must be fulfilled before execution of this test case. List all the pre-conditions in order to execute this test case successfully.

**Test Steps:** List all the test execution steps in detail. Write test steps in the order in which they should be executed. Make sure to provide as many details as you can.

**Test Data**: Use of test data as an input for this test case. You can provide different data sets with exact values to be used as an input.

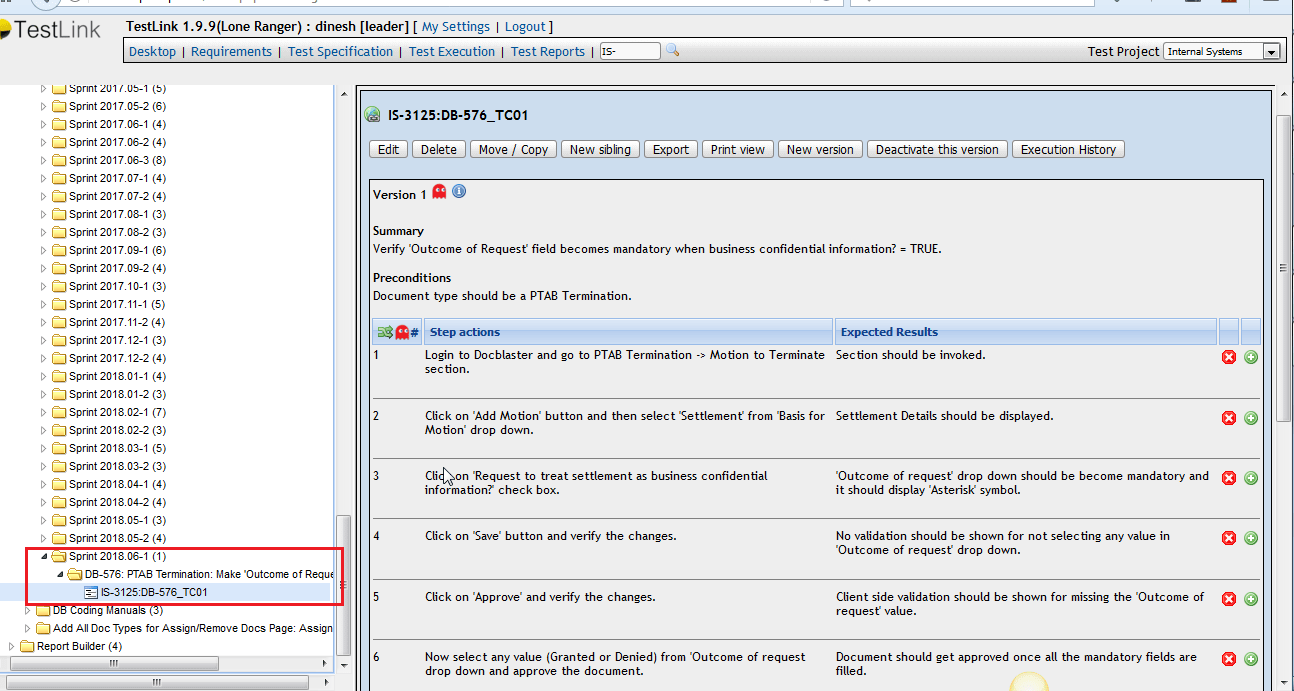
**Expected Result:**  What should be the system output after test execution? Describe the expected result in detail including message/error that should be displayed on the screen.

**Post-condition:** What should be the state of the system after executing this test case?

**Actual result:** Actual test result should be filled after test execution. Describe system behaviour after test execution.

**Status (Pass/Fail):** If actual result is not as per the expected result mark this test as failed. Otherwise, update it as passed.

***Test Case Template:***



**17.Test management tool**

Test management tools are used to store information on how testing is to be done, plan testing activities and report the status of quality assurance activities.

The tools have different approaches to testing and thus have different sets of features.

They are used to maintain and plan manual testing, run or gather execution data from automated tests, manage multiple environments and to enter information about found defects.

**18.Requirements management tool**

**Requirements** management is the process of documenting, analyzing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders.

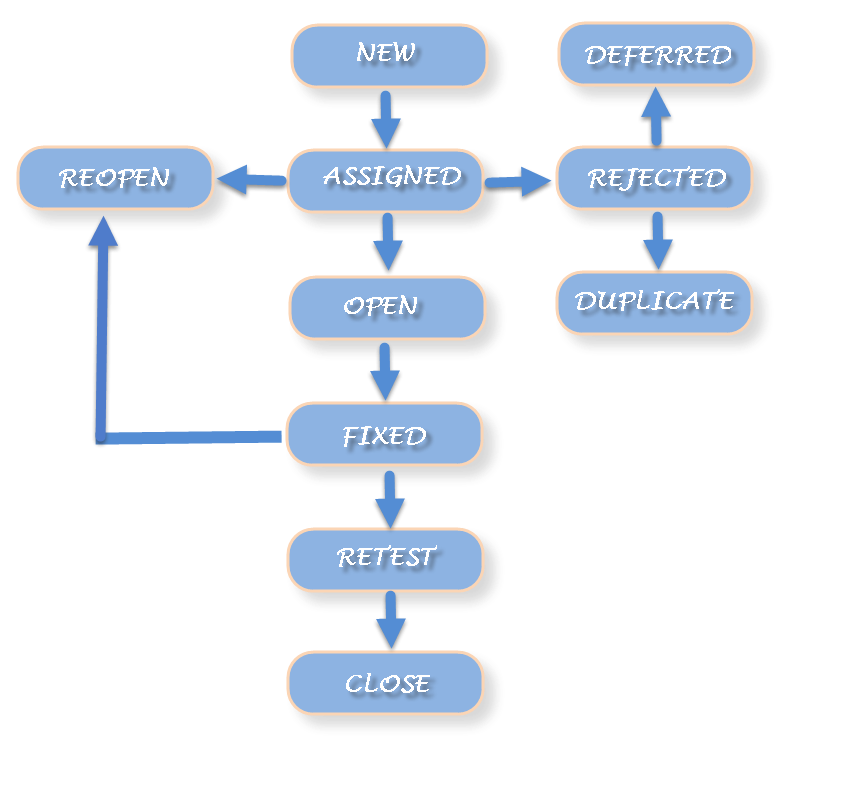
Requirements Management Tool is a software system that helps you manage the various manually intensive tasks in the requirement development and Requirements management process

**18.Defect tracking tool**

A bug tracking system or defect tracking system is a software application that keeps track of reported software bugs in software development projects.

**19.Bug life cycle**

Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime.



**New:** When a defect is logged and posted for the first time. Its state is given as new.

**Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to developer team.

**Open:** Its state when the developer starts analyzing and working on the defect fix.

**Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’.

**Retest:** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

*Once the latest build is pushed to the environment, Dev lead move all the Fixed defects to Retest. It is an indication to the testing team that the defects are ready to test.*

**Reopened:**  If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”.

**Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases.

**Rejected:** If the developer feels that the bug is not genuine, developer rejects the bug.

**Duplicate :** If the bug is repeated twice or the two bugs mention the same concept of the bug, then the recent/latest bug status is changed to “duplicate“.

**Closed:** If the tester feels that the bug no longer exists in the software, tester changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.

**20.Information while creating a bug**

**Defect ID**

**Title/Summary**

**Defect Reported Date**

**Environment**

**Status**

**Description**

**Pre-requisites**

**Steps to reproduce**

**Expected Result**

**Actual Result**

**Screenshot**

**Defect ID :**The Defect ID will be generated automatically in case of defect management tool.

**Title/Summary:** Title should be short and simple. It should contain specific terms related to the actual issue.

**Defect Reported Date:** Mention the date on which you have found the bug.

**Environment:** In which environment in which you have encountered the bug.

**Status:** Specify the status of the bug.

(E.g. New/ Assigned/ Open/ Fixed/ Test/ Verified/ Closed/ Reopen/ Duplicate/ Deferred/ Rejected/ cannot be fixed/ Not Reproducible/ Need more information)

**Description :**In the description section, you must briefly explain what you have done before facing the bug.

**Prerequisites:** What should be the state of the system after executing this test case?

**Steps to reproduce :**In this section, you should describe how to reproduce the bug in step by step manner.

These steps should describe the bug well enough and allows developers to understand and act on the bug

**Expected Result:**

What is the expected output from the application when you make an action

**Actual Result:**

What is the expected output from the application when you make an action

**Screenshot:** Screenshot of the test case created.