**MANUAL TESTING**

* **In Manual Testing test cases are executed manually by testers, without using any automated tools.**
* **The purpose of manual testing is to identify the bugs, issues, and defects in the software application.**
* **What is Testing: Checking something how well it works.**
* **WHAT IS SOFTWARE TESTING: To check whether software is doing what it is supposed to do and not doing what it is not supposed to do.**
* **WHY TO TEST: To avoid risk, error free, secure, efficient etc.**

GOAL OF MANUAL TESTING/ SOFTWARE TESTING

* **To make sure that the application under test is defect free and the application is working as per the requirement specification document.**

**TYPES OF MANUAL TESTING**

* **In fact, any type of software testing type can be executed both Manually as well as using an Automation Tool.**
* **Black box testing**
* **White box testing**
* **Unit testing**
* **Integration testing**
* **System testing**
* **Acceptance testing**

**BLACK BOX TESTING**

* **Black box testing is testing the functionality of an application with respective customer requirements. Done by the test engineers.**
* **Tools – Selenium, Appium, Applitools, HP QTP, Microsoft coded UI.**

**TYPES OF BLACK BOX TESTING**

1. **Functional testing – is related to functional requirements of a system and covers how well the system executes its functions. (According to customer requirements)**
2. **Non-Functional testing – is related to non-functional requirements** like **(performance, compatibility, reliability, etc)**
3. **Regression testing – It is done after code fixes or changes or system upgrades to ensure that any code modification should not impact the existing functionalities of the software.**

**WHITE BOX TESTING**

* **White box testing is testing the internal logic of the program to improve Design, Usability and Security.**
* **It is also called Clear box testing, Open box testing, Code-based testing and Glass box testing.**

**TYPES OF WHITE BOX TESTING**

1. **Unit testing**
2. **Integration testing**
3. **Regression testing**

WHITE BOX TESTING TECHNIQUES

* + - 1. **Mutation testing**
      2. **Static code analysis**

**3. Dynamic code analysis**

**4. Statement coverage**

**5. Branch coverage**

**6. Path coverage, etc.**

* **GREY BOX TESTING - Combination of Black box and White box testing**

**level of Testing:**

1. **UNIT TESTING (D) – Testing a single component or single program or single module is called unit testing. A unit is a smallest testable piece of the software (code).**

**Purpose: To check the control flow and data flow of the application.**

1. **INTEGRATION TESTING (D) – Testing the connection between two or more modules, or group of interacting modules is tested together.**

**There are 4 types of Integration testing:**

* **Top down**
* **Bottom up**
* **Hybrid/ Sandwich**
* **Big-Bang**

1. **SYSTEM TESTING (T) – Testing the complete software product with respective customers’ requirements. [Comes under black box testing].**
2. **ACCEPTANCE TESTING UAT (C) – Testing a system for Acceptability. This testing is done to check whether the system has met the customer requirement specifications or not.**

**Acceptance test is basically done by the users or customers.**

**[Acceptance test conduct in 2 levels – Alpha testing and Beta testing].**

**Alpha:- Done by Testers Beta:- Done by Customers/ Clients**

**SOFTWARE TESTING LIFE CYCLE (STLC)**

* **Requirements Analysis: [Analyzing, read, and understanding SRS - system requirement specification]**
* **Test planning: [How testing will be carried, test plan document, what to test, when to test]**
* **Test cases development: [HLD – Broder view – Test scenarios. LLD – Detailed view – Test cases (how to test). Black box and white box test design techniques are used to design the test cases for testing.]**
* **Test Execution:** [**Executing test cases, Identifying defects.]**
* **Defect Reporting: [Reporting the defects to the developer and in bug tracking tool.]**
* **Test Cycle Closer: [Analyzing Test Reports, post implementation reviews, prepare of test closure report.]**

**SOFTWARE DEVELOPMENT LIFE CYLE (SDLC)**

* **Requirement gathering**
* **Designing**
* **Building**
* **Testing**
* **Deployment**
* **Maintenance**

**SMOKE AND SANITY TESTING/ functional**

* **SMOKE TESTING – It is initial level of testing mainly focuses on Installation and stability. Also known as BVT – build verification test.**
* **SANITY TESTING – Testing the basic functionalities is working or not and checking the behavior of the application. Example: page is displayed or not, login is working or not, all the elements are displaying or not.**

**LEVEL OF TESTING IN DEPTH**

* **UNIT TESTING - [mostly done by the developers] - W**
* **INTEGRATION TESTING - [mostly done by the developers] - W**
* **SYSTEM TESTING (main) - [done by the testers] -B**
* **UAT TESTING - [done by the customers\users]**

**SYSTEM TESTING**

**System Testing includes both Function testing as well as non Function Testing**

* **GRAPHICAL USER INTERFACE: (GUI) TESTING**
* **GUI testing is testing the basic elements are available in the page or not, spelling of the object is correct or not, alignment of the object is correct or not, all controls like menus, icons dialog boxes are available or not, etc.**
* **The main focus of the GUI testing is (look and feel of the application) like how much our application is attractive to users.**
* **USABILITY TESTING: – (How easy to use testing)**
* **Testing how easily the end users are able to understand and operate the application**
* **BLACK BOX TEST DESIGN TECHNIQUES/ FUNCTIONAL TESTIING**

1. **Object properties coverage**
2. **Input domain coverage (BVA, ECP)**
3. **Database testing/backend coverage**
4. **Error handling coverage**
5. **Calculation and manipulation coverage**
6. **Links existence and links execution**
7. **Cookies and sessions, etc.**

* **OBJECT PROPERTIES TESTING:**
* **During functional testing testers validates properties of objects in run time. Every object has certain properties such as enable, disable, focus, text, height, weight and so on.**

**EXAMPLE: - This is object and the property of this object is Enable or disable, size, accepting the text or not etc.**

* **INPUT DOMAIN TESTING:**
* **Input domain testing is testing a minimum and maximum numbers of inputs to check the output of a system**
* **The purpose of IDT is to check the system that accepts only certain inputs within the acceptable range or not.**

**Example: - This textbox only allows 4-8 characters, minimum 4 and maximum 8. And textbox allows only lowercase letters a-z.**

* **There are two techniques in IDT – (ECP and BVA)**
* **Equivalent Class Partitioning (ECP)** – Checks Valid, Invalid.
* **ECP** – Technique is used to divide the input data of software into different equivalence data classes. ECP - is a testing technique where input values set into classes for testing.

Example - A text field permits only numeric characters. Length must be 6-10 characters long. So partition according to the requirement should be like this.

|  |  |  |
| --- | --- | --- |
| **Invalid** | **Valid** | **Invalid** |
| 0,1,2,3,4,5 | 6,7,8,9,10 | 11,12,13,14,15 |

* **Boundary Value Analysis (BVA)** – Checks Boundary Values.
* **BVA** - Technique is used to test Boundary value between partitions (both valid Boundary partition and Invalid partition). It is a part of Stress and Negative Testing.

**EXAMPLE**: Text box accepts minimum - 8 characters and maximum -12. Valid range is 8 – 12, and Invalid range is 7 or less than 7 and 13 or more than 13.

|  |  |  |
| --- | --- | --- |
| **Invalid** | **Valid** | **Invalid** |
| Less than 8 | 8 – 12 | More than 12 |

* **DATABASE TESTING:**
* **Database testing validates DML (data manipulation language) operations like Insert, Update, Delete, Select, etc.**
* **SQL Languages such as DDL, DML, DCL etc. this test is done by database testers especially DDL and DCL who knows SQL languages.**

**DDL – Data definition language – (create, alter, and drop)**

**DML – Data manipulation language – (insert, update, select, delete)**

**DCL – Commit, roll back, etc.**

* **ERROR HANDLING TESTING:**
* **Testing error messages thrown by the application or from text boxes when we input invalid data.**

**Example: when we input wrong data it should throw certain message to the user like password wrong, password invalid etc. and application should throw meaningful message which should be understandable by the user.**

* **CALCULATION/MANIPULATIONS TESTING:**
* **All types of calculation are checked like addition, subtraction, multiplication etc.**

**Example: you have done some fund transfer to other person and as soon as you have sent fund that fund will be minus from your existing balance and that should we updated proper.**

**Example: as soon as we debit from ac the amount should be reduced from the balanced. As soon as we credit to ac the amount should be added in balance. This is what we must check.**

* **LINK COVERAGE TESTING:**
* **Link placed in the appropriate location or not. Link is navigating to appropriate page or not.**
* **Types of links: there are 3 types of links in web pages.**
* **Internal links = navigates in same page.**
* **External links = navigates in next pages or targeted pages.**
* **Broken links = don’t have any target pages, it is made for future purpose for implement.**
* **COOKIES AND SESSIONS TESTING:**
* **A cookie is a small amount of data generated by a website and saved by your web browser. Its purpose is to remember information about you. Cookies are created at client side.**
* **Most common purpose of cookie is to store login information for a specific site. Example: Facebook, Instagram etc.**
* **SESSIONS:**
* **Sessions are time slots which are allocated to the user at the server side.**

**Example: suppose you login a page and not using for 5 to 10 minutes and when you try to use a page after 10 min, automatically page will ask to login again. It is for security purpose usually used in banking systems.**

* **NON-FUCTIONAL TESTING:**
* **Nonfunctional testing is done by other specialist testers by using tools like Jmeter, LoadRunner, etc.**

1. **Performance testing - (web application)**

* **Load testing**
* **Stress testing**
* **Volume testing**

1. **Security testing**
2. **Recovery testing**
3. **Compatibility testing**
4. **Configuration testing**
5. **Installation testing**
6. **Sanitation testing**

* **PERFORMANCE TESTING: (is only done for web-based application)**
* **Load testing: (Expected Load)**
* **To test the performance or behaviour of the system under an expected load**.

**Testing the performance of an application with max users at same time.**

**Example: first we login system with 10 people, then 20, then 30, etc. and see application is stable or not when increasing load.**

* **Stress testing: (Extreme Load)**
* **Help to understand the upper limits of a system's capacity using a load beyond the expected maximum. It helps to determine how a system would behave under an extreme load**.

**Testing the speed of the system by increasing load and suddenly reducing the load in the system to check anywhere its breaking or not.**

**Example: login system with 10 people then increasing with 50 and again reducing 10, then 20, then 90, etc. to see the system is stable or not.**

* **Endurance testing:**  **It is done to make sure the software can handle the expected load over a long period of time.**
* **Spike testing: This tests the software’s reaction to sudden large spikes in the load generated by users**.
* **Volume testing**: **Under Volume Testing large no. of. Data is populated in a database and the overall software system’s behaviour is monitored. The objective is to check the software application’s performance under varying database volumes.**
* **Volume testing: check how much volume of data can handle by the system. Also known as storage testing or memory testing.**
* **Scalability testing:** **The objective of scalability testing is to determine the software application’s effectiveness in scaling up to support an increase in user load. It helps plan capacity addition to your software system.**
* **SECURITY TESTING:**
* **It is a type of testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attack from intruders.**
* **The purpose of security test is to identify all possible loopholes and weakness of the software system.**
* **It is done by security testing specialist because it required skills for testing with the help of tools like Intruder, Acunetix, Owasp, Wireshark, etc.**
* **RECOVERY TESTING:**
* **Testing how well system recovers from crashes, hardware failure or sudden problems.**

**Example: suppose you are working in some program and suddenly light goes off then computer should be able to recover those data**

* **COMPATIBILITY TESTING:**
* **Testing compatibility of the system like Operating system, hardware and browsers.**

1. **Operating system compatibility – checking product is installable or working or not in other operating systems like, Linux, UNIX and windows, etc.**
2. **Hardware compatibility (configuration testing) – suppose customer have 2GB ram in his system and 500 GB hard disk and AMD processor. So whatever application we are developing that application should work in this environmental configuration.**
3. **Browser compatibility – checking we based application is working or accessing all browser or not like chrome, Firefox, internet browser, opera browser, etc.**
4. **Forward and backward compatibility testing – checking whether our application is working or supporting in latest version or coming future version of the operating system is known as forward compatibility testing. Example: latest version of OS is windows 10 and coming future versions like 11, 12, 13, 14, etc.**

**Backward compatibility – checking whether application is supporting the older versions of OS is known as backward compatibility. Example: windows 7, windows XP, etc.**

* **INSTALLATION TESTING:**
* **Testing installation of the application on customer expected platforms and checking installation steps, navigation while installing, and how much space is occupied in memory.**
* **Also we need to test uninstallation process. Whenever we do uninstallation each and every files which belong to the software should be remove from system.**
* **SANITATION/ GARBAGE TESTING:**
* **Testing the application is providing additional or extra functionalities that is called garbage testing.**
* **Application is providing extra functions which is not part of requirements is garbage testing or sanity testing. It is a subset of regression testing.**

**TESTING TERMINOLOGY: (imp for interview)**

* **ADHOC TESTING: - Randomly testing the functionality by knowing the functionality without planning, without following specifications and documentation.**
* **MONKEY TESTING: - Due to lack of time, the testing team concentrates on some of the main activities in the software build for testing. This is known as monkey testing or chimpanzee testing or gorilla testing.**
* **RETESTING: - Testing functions repeatedly again and again is called re-testing.**

**OR**

**Testing same functions again and again but using different types of input is called re-testing.**

* **REGRESSION TESTING: - Is testing after code fixes/changes or system upgrades to ensure that any code modification should not impact the existing functionalities of the software.**
* **END TO END TESTING: - Testing the complete or overall functionalities of the system is called end to end testing.**
* **EXPLORDATORY TESTING: - Exploring the application and understanding the functionalities for better testing is called exploratory testing. It is perform when test cases are not ready then we do exploratory testing especially for new joiners.**
* **GLOBALIZATION TESTING OR INTERNATIONALIZATION TESTING: (I18N) – Checks if the application has a provision of setting and changing languages, date and time format and currency etc. if it is designed for global users, it is called globalization testing.**
* **LOCALIZATION TESTING: - Checking default languages, currency, data and time format etc. if it is designed for a particular locality of users is called localization testing.**
* **POSITIVE TESTING (+VE): - Testing application in a positive approach to determine what system is supposed to do is called a positive testing.**
* **NEGATIVE TESTING (-VE): - Testing a software application with a negative perception to check what system is not supposed to do is called negative testing.**

**STLC (IN DEPTH)**

1. **REQUIREMENT ANALYSIS – read and understand SRS**
2. **TEST PLANNING - how testing will be carried out, test plan document**
3. **TEST DISIGNING (preparation of test scenarios and test cases)**
4. **TEST EXECUTION – post implementation reviews, actual and expected result**
5. **IDENTIFY DEFECTS AND BUG REPORTING**
6. **TEST CLOSURE- preparation of test closure report**

* **REQUIREMENT ANALYSIS (Gathering):**
* **It is a process of defining user expectations for a new software being built or modified. Requirement analysis is a detail description of the system under implementations.**
* **TEST PLANNING: (In depth)**
* **Test plan is a document describing the scope, approach, resources and schedule of testing activities.**
* **Test plan contents – features to be tested, features not to be tested, entry criteria, exit criteria, test approach (strategy), test environment, risk and mitigations etc. test plan is prepared by senior people like managers test leaders etc.**
* **TEST DESIGNING: (preparation on test scenarios, test cases, etc.)**
* **Test designing is the activity of deriving and specifying test cases from test conditions to test software.**

1. **USE CASE – Use case is used to define the system that how to use the system for performing a specific task. Use case describe the requirements.**

**A use case is not part of execution it is only a diagrammatic presentation of a document that specifies how to perform a certain task.**

**Use case contains 3 items = actor, action/flow, outcome.**

1. **TEST SCENARIOS – It describes what to test, that are the different areas to be tested. Test scenarios are derived from requirements.**

**Example: validation of login functionality of Gmail application.**

1. **TEST CASE – It describes how to test, those scenarios step by step process in detail. Test cases are derived from test scenarios.**

**Example: checking the functionality of login button.**

**TC1 – click the button without entering user name and password.**

**TC2 – click the button only entering user name.**

**TC3 – click the button while entering wrong user name and password.**

* **TEST CASE DESIGN TECHNIQUES: (also known as black box testing techniques)**

1. **Boundary value analysis (BVA)**
2. **Equivalence class partitioning (ECP)**
3. **Decision table testing technique**
4. **State transition diagram technique**
5. **Use case testing technique**
6. **Error guessing technique**

* **Boundary Value Analysis (BVA)**
* **BVA technique is used to check the error at the boundaries of an input domain. This technique is used to test boundaries value between partitions (both valid boundary partitions and invalid partitions). It is a part of stress and negative testing.**
* **Equivalence Class Partitioning (ECP)**
* **ECP technique is used to divide the input data of software into different equivalence data classes.**
* **Decision Table Testing Technique**
* **It is used to test system behaviour for various input combinations. In this approach, the several input combinations and their corresponding system behaviour are represented in Tabular form.**

**EXAMPLE**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EMAIL** | V | V | I | B | B | I |
| **PASSWORD** | V | I | V | V | B | I |
| **Expected Result** | Passed | Failed | Failed | Failed | Failed | Failed |

**V - Valid and I – invalid**

* **State Transition Testing**
* **It is performed to check the change in the state of the application under varying input. In this type of testing both positive and negative inputs values are provided and the behaviour of the system is observed**.

**EXAMPLE: Read manual testing book.**

* **Use Case Testing**
* **Use case technique helps to identify test cases that cover the entire system, on a transaction-by-transaction basis from start to finish.**
* **Error Guessing Technique**
* **Error guessing technique is guessing the error which can prevail in the code. It is experience based testing technique where the test analyst uses his experience to guess the problematic areas of the application. This technique necessarily requires skilled and experienced testers.**

**\*Key points\***

* **Test Suit: – It is group of test cases which belongs to same category to show that it has some specified set of behaviour. For example read manual testing book.**
* **Test Case Content:**
* **Test case ID**
* **Test case Tittle**
* **Description**
* **Pre-condition**
* **Priority (P0, P1, P2, P3, etc.)**
* **Requirements**
* **Steps/Actions**
* **Expected Result**
* **Actual Result**
* **Test data**
* **Characteristics of Good Test Case:**
* **No unnecessary steps should be included in it.**
* **It should be traceable to requirements.**
* **It should be reusable.**
* **It should be accurate and test what it is intended to test.**
* **It should be simple and clear; any tester should be able to understand it by reading once.**
* **Positive v/s Negative Test Cases: read from manual book.**
* **TEST EXECUTION: (imp)**
* **During test execution testing team will carry out the testing based on the test plans and the test cases which are already prepared.**
* **Bugs will be reported back to the development team for correction and retesting will be performed.**
* **DEFECT REPORTING or BUG REPORTING:**
* **Any mismatched functionality found in an application is called as defect/bug/issue.**
* **During test execution test engineers are reporting mismatches as defects to developers through templates or using tools.**
* **Defect Reporting Tools:**
* **Clear quest**
* **Dev track**
* **Jira**
* **Quality centre**
* **Bug Jilla etc.**
* **Defect Report Contents:**
* **Defect ID – Unique identification number for the defect.**
* **Defect Description – Detailed description of the defect including information about the module in which defect was found.**
* **Version – Version of the application in which defect was found.**
* **Steps – Detailed steps along with screenshots with which the developer can reproduce the defects.**
* **Data Raised – Data when the defect is raised.**
* **Reference – Where in you provide reference to the document like requirements, design, architecture or may be even screenshots of the error to help understand the defect.**
* **Defected By – Name/ID of the tester who raised the defects.**
* **Status – Status of the defect, more on this later.**
* **Fixed By – Name/ID of the developer who fixed it.**
* **Date closed – Date when the defect is closed.**
* **Severity – which describes the impact of the defect on the application.**
* **Priority – which is related to defect fixing urgency. Severity priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively.**
* **Severity – (seriousness of the defect, how impact it is)**
* **Priority – (how much early the defect should be fixed)**
* **Defect Management Process:**
* **Defect Classification:**

**[Defect Categorization]**

|  |  |
| --- | --- |
| **SEVERITY** | **PRIORITY** |
| **Critical** | **P1** |
| **High** | **P2** |
| **Medium** | **P3** |
| **Low** | **P4** |

* **Severity always defined by the testers.**
* **Priority is decided by developer and Business Analysts.**
* **Critical means showstopper, Blocker, etc.**
* **P0 and P1 is a Blocker and showstopper.**
* **Defect severity:**
* **Severity describes the seriousness of defect.**
* **Defect severity can be categorized in to 4 classes.**

1. **Critical – Indicates complete shut-down of the process, nothing can proceed further.**
2. **High – It is highly severe defect and collapse the system. However, certain parts of the system remain functional.**
3. **Medium – It cause some undesirable behavior, but the system is still functional.**
4. **Low – It won’t cause any major break-down of the system.**

* **Defect Priority:**
* **Priority describes the importance of defect. Priority describes how much early the defects should be fixed.**
* **Defect priority can be categorized into 3 class.**

1. **P1 – The defect must be fixed as soon as possible as it affects the system severely and cannot be used until it is fixed.**
2. **P2 – During the normal course of the development activities defects should be resolved or fixed. It can wait until a new version is created.**
3. **P3 – The defect is an irritant, but repair can be done once the more serious defect have been fixed.**

* **LS/HP and HS/LP: – (examples imp)**

1. **Low severity with High priority:**

* **A logo error for any shipment website, can be of low severity as it not going to affect the functionality of the website. But can be of high priority as you don’t want any further shipment to proceed with wrong logo.**

1. **High severity with a Low priority:**

* **For a flight operation website, defect in reservation functionality may be of high severity, but can be low priority as it can be scheduled to release in a next cycle.**

1. **High priority and High severity defect:**

* **If login is required for an application and the users are unable to login to the application with valid user ID. Such defects need to be fixed with high importance. Since it is stopping the customer to progress further.**

1. **Low priority Low Severity:**

* **A spelling mistake in a page not frequently navigated by users.**

1. **High priority High severity:**

* **Issue with login functionality.**

1. **High priority Low severity:**

* **Defect Resolution: (it’s an opinion by the developer what to do further)**
* **After receiving the defect report from the testing team, development team conduct a review meeting to fix defects. They send a Resolution type to the testing team for the further communication.**
* **Resolution Types:**
* **Accept**
* **Reject**
* **Duplicate**
* **Enhancement**
* **Need more information**
* **Not Reproducible**
* **Fixed**
* **As designed**
* **Defect Triage:**
* **Defect triage tries to do the re-balancing of the process where test team faces the problem of the limited availability of resources.**

**Example: when there are large number of defects and limited testers to verify them, defect triage helps trying to get as many defects resolved based on defect parameters like severity and priority.**

* **Defect Triage process:**
* **Defect Review**
* **Defect Assessment**
* **Defect Assignment**
* **Tips for Good Bug Report:**

1. **Structure: - Test carefully (use deliberate, careful approach to testing)**
2. **Reproduce: - Test it again (rule of thumb 3 times)**
3. **Isolate: - Test it differently (change variables that my alter symptoms)**
4. **Generalize: - Test it elsewhere does the same failure occur in other modules or locations?**
5. **Compare: - Review results of similar test (same test run against earlier versions)**
6. **Summarize: - Relate test to customers (put a short “tagline” on each report)**
7. **Condense: - Trim unnecessary information (Eliminate extraneous words or steps)**
8. **Disambiguate: - Use clear words (Goal: clear, indisputable statements of fact)**
9. **Neutralize: - express problem impartially (deliver bad news gently)**
10. **Review: - Be sure (peer review)**

* **Bug Life Cycle: (diagram)**
* **Bug life cycle or Defect life cycle is a process in which defect goes through different stages in its entire life.**
* **This life cycle starts as soon as a bug is reported by the testers and ends when testers ensures that the issue is fixed and won’t occur again. For Diagram read manual testing book.**
* **States of Defects:**
* **State: - It is first state of the lifecycle in which the bug is identified. Whenever new bug is found, it is assigned a new state.**
* **Assigned: - Assigned to developer to fix the bug.**
* **Open: - when developer starts working on the bug, it falls in an open state.**
* **Fixed: - when bug is fixed and waiting for validation.**
* **Closed: - The bug is fixed, validated and closed.**
* **Rejected: - Bug is not genuine.**
* **Deferred: - Fix will be placed in future builds.**
* **Duplicate: - Two bugs mention the same concept.**
* **Need more Info: - when the developer needs information to reproduce the bug or to fix the bug.**
* **Invalid: - Not a valid bug.**

**Reopened: - when bug still exits even after the bug is fixed by the developer.**

* **TEST CYCLE CLOSURE:**
* **Test cycle closure is a document that gives a summary of all the tests conducted during the software development life cycle.**
* **It also gives a detailed analysis of the bugs removed and errors found.**
* **Activities: - we need to identify all the test cases are executed and passed or not, all the defects are being closed or not and then we can exit from the testing.**
* **Deliverables: - Test closure report, Test metrics, etc.**

**MIND SCRIPTS NOTES:**

1. **What is Defect?**

* **Any difference between Expected and Actual Result.**

1. **Who reports defect?**

* **Tester and the clients who are using the application can report defect.**