CHAPTER 1
INTRODICTION TO SOFTWARE TESTING

SDLC(Software Development Lifecycle):-

It is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customers need and expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase.

SDLC phases are:-

1. **Requirement Gathering and Analysis**
2. **Design**
3. **Coding**
4. **Testing**
5. **Installation/Deployment**
6. **Maintenance**

CODING:-

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process. Coding, is the phase when your team works on the actual software and translates software specs and requirements into computer code. Physical software specifications are transformed into a solid working and reliable solution. Developers(coder/programmer) collect and create source code and BUILD(.exe file)

TESTING:-

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality and Non-functionality of the entire system. This is done to verify that the entire application works according to the customer requirement. During this phase, testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug /defect and send back to testers for a re-test. This process continues until the software is defect/bug-less, stable, and working according to the business needs of that system. Here Testers create several documents such as:-

i. Test Plan
ii. Test Scenario
iii. Test Case
iv. Defect Report
v. Test Execution Log
vi. Test Metrics
vii. Test Suites
viii. Test Procedure
ix. Test Execution Schedule
x. Test Summary Report
xi. Test Progress Report etc.....

Installation/Deployment:-

Once the s/w testing is been completed, the tested s/w is installed at the client/customer side (i.e at the client environment) and this is what happening in this phase.
Installation is done by the installation Team/Deployment team. This team is created by the project manager which consists of selected developers and testers . This team creates a documents called as User Manual/User Guide which is provided to customer for the understanding purpose of the s/w

Maintainence:-

Once the system is deployed, and customers start using the developed system, and there may occurs some issues/changes such as:-there are 2 types of changes

1) **Adaptive change:**
    a. Enhancement - Adding some new features into the existing software
2) **Corrective changes:**
    a. Bug fixing/defect fixing - bugs are reported because of some scenarios which are not tested at all
    b. Upgrade /modification- Upgrading the application to the newer versions of the Software/ modifying the existing feature in the application
    c. Removing- Deleting some existing feature in the s/w

Maintenance is done by **Maintenance team/Configuration Management team.** This team is created by the project Manager which consists of project manager himself along with BA, S/W designer, Developer, testers etc..

Different SDLC Models:-

1. Waterfall Model:-

(a) Introduction:-

- It is common and classical Life cycle model.
- The Waterfall Model was first Process Model to be introduced.
- It is also referred to as a linear-sequential life cycle model.
- It is very simple to understand and use.
- Detailed documentation in each phase before proceeding to next phase.
- Project is completely dependent on the members of team rather than client feedback.
- At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project.

(b) When to use the waterfall model

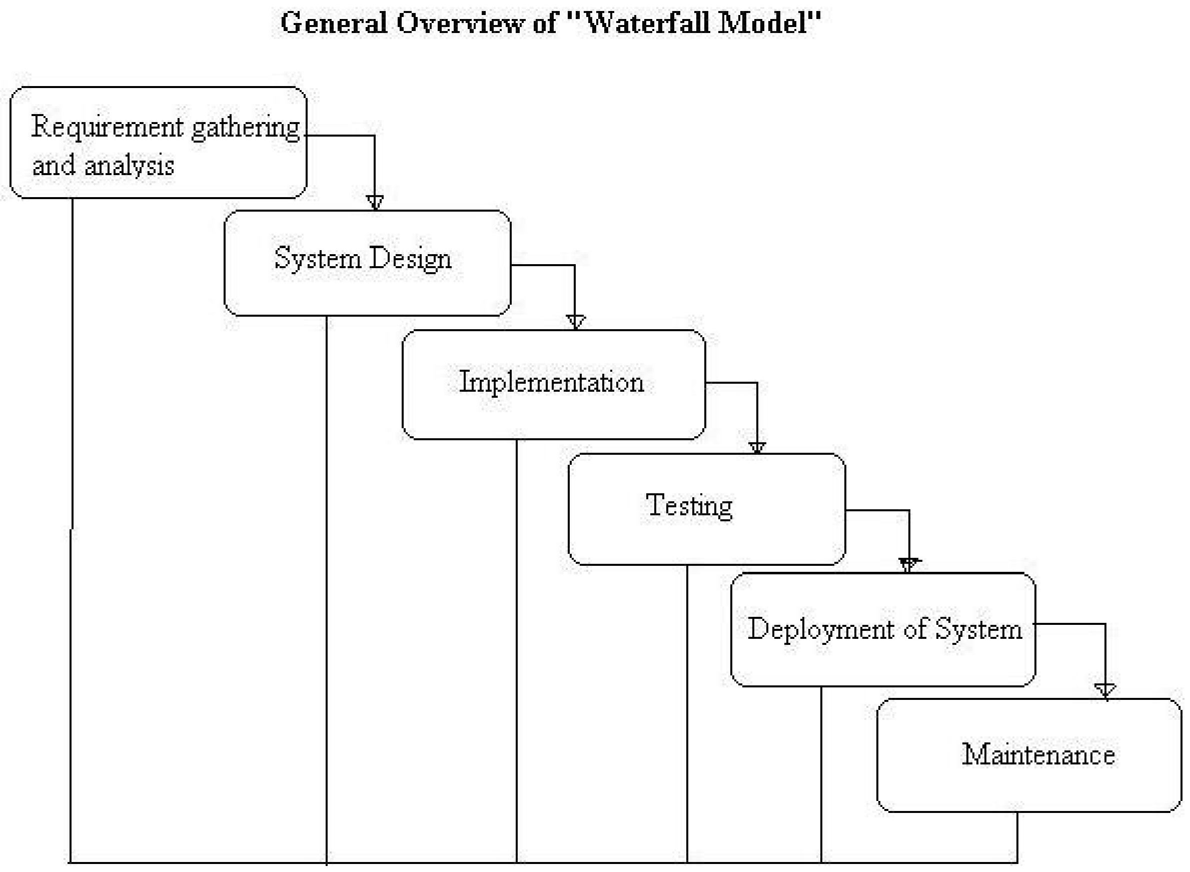
- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements.
- Ample resources with required expertise are available freely.
- The project is short.

(c) Advantages of waterfall model

- Simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specifc deliverables and a review process.
- It works well for smaller projects where requirements are clearly defned and very well understood.

(d) Disadvantages of waterfall model

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well in the previous stages.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing and complex and object-oriented projects.


General Overview of "Waterfall Model"
Requirement gathering and analysis
System Design
Implementation
Testing
Deployment of System
Maintenance

2. Incremental Model (Multi-Waterfall Model):-

(a)Introduction:-

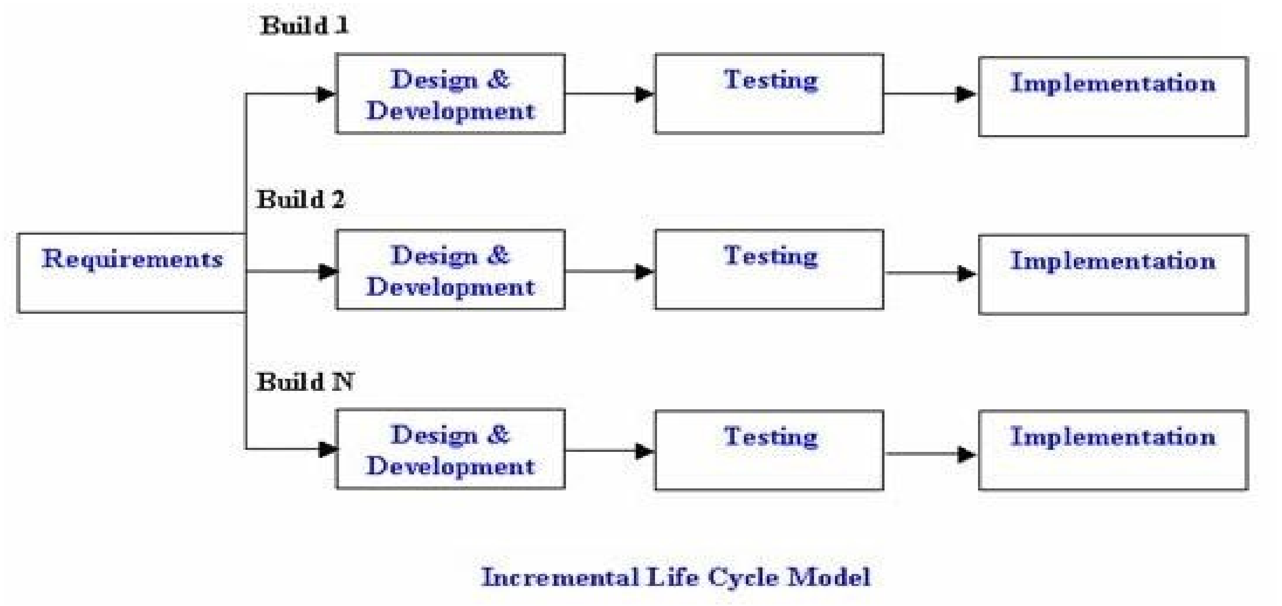
- In incremental model the whole requirement is divided into various builds.
- Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle.
- Cycles are divided up into smaller, more easily managed modules.
- In this model, each module passes through the requirements, design, implementation and testing phases.
- It doesn't at least start with the full specification of requirement.
- Instead development begin by specifying and implementing just as per high priority requirement of the s/w.
- A working version of software is produced during the first module, so you have working software early on during the software life cycle.
- Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

(b) <u>Advantages of Incremental model</u>

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it's iteration.

(c) <u>Disadvantages of Incremental model</u>

- Needs good planning and design.
- Needs a clear and complete defnition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.



3. <u>Spiral Model :-</u>

(a)<u>Introduction:-</u>

- The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation.
- A software project repeatedly passes through these phases in iterations (called Spirals in this model).
- The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed.
- Each subsequent spirals builds on the baseline spiral.

(b) <u>Advantages of Spiral model</u>

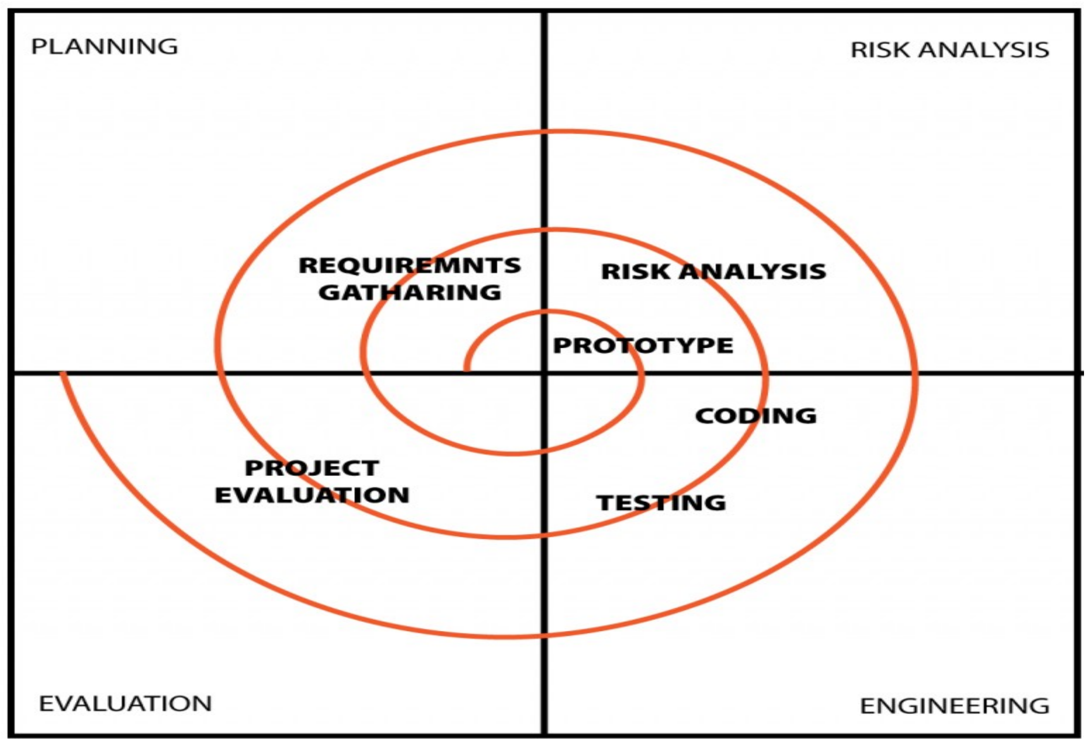
- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

(c) <u>Disadvantages of Spiral model</u>

- Can be a costly model to use.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

(d) <u>When to use Spiral model</u>

- When costs and risk evaluation is important
- For medium to high-risk projects
- Users are unsure of their needs
- Requirements are complex
- Significant changes are expected.



AGILE:-

- Agile is an umbrella term for several iterative and incremental software development approaches, with each of those variations being its own Agile framework. (Framework means a set of protocol defned for doing a method and process)
- Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality.
- Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.
- Extreme Programming (XP) is currently one of the most well known agile development life cycle model
- The most popular Agile frameworks include Kanban, Scrum, Extreme Programming etc.

➔ **Manifesto for Agile Software Development**

We are uncovering better ways of developing software by doing it and helping others do it.Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.

- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- It is not useful for small development projects.
- There is a lack of intensity on necessary designing and documentation.
- It requires an expert project member to take crucial decisions in the meeting.
- Cost of Agile development methodology is slightly more as compared to other development methodology.

When to use Agile model:-

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the waterfall model in agile model very limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

AGILE- Scrum Methodology :-

Scrum:-

What Are the Components of Agile Scrum Development?

The Scrum methodology is defined by team roles, events (ceremonies), artifacts, and rules.

1. **The Scrum Team:-** Scrum teams are typically composed of 7 +/- 2 members and have no team leader to delegate tasks or decide how a problem is solved. The team as a unit decides how to address issues and solve problems. Each member of the Scrum team is an integral part of the

solution and is expected to carry a product from inception to completion. There are three key roles in a

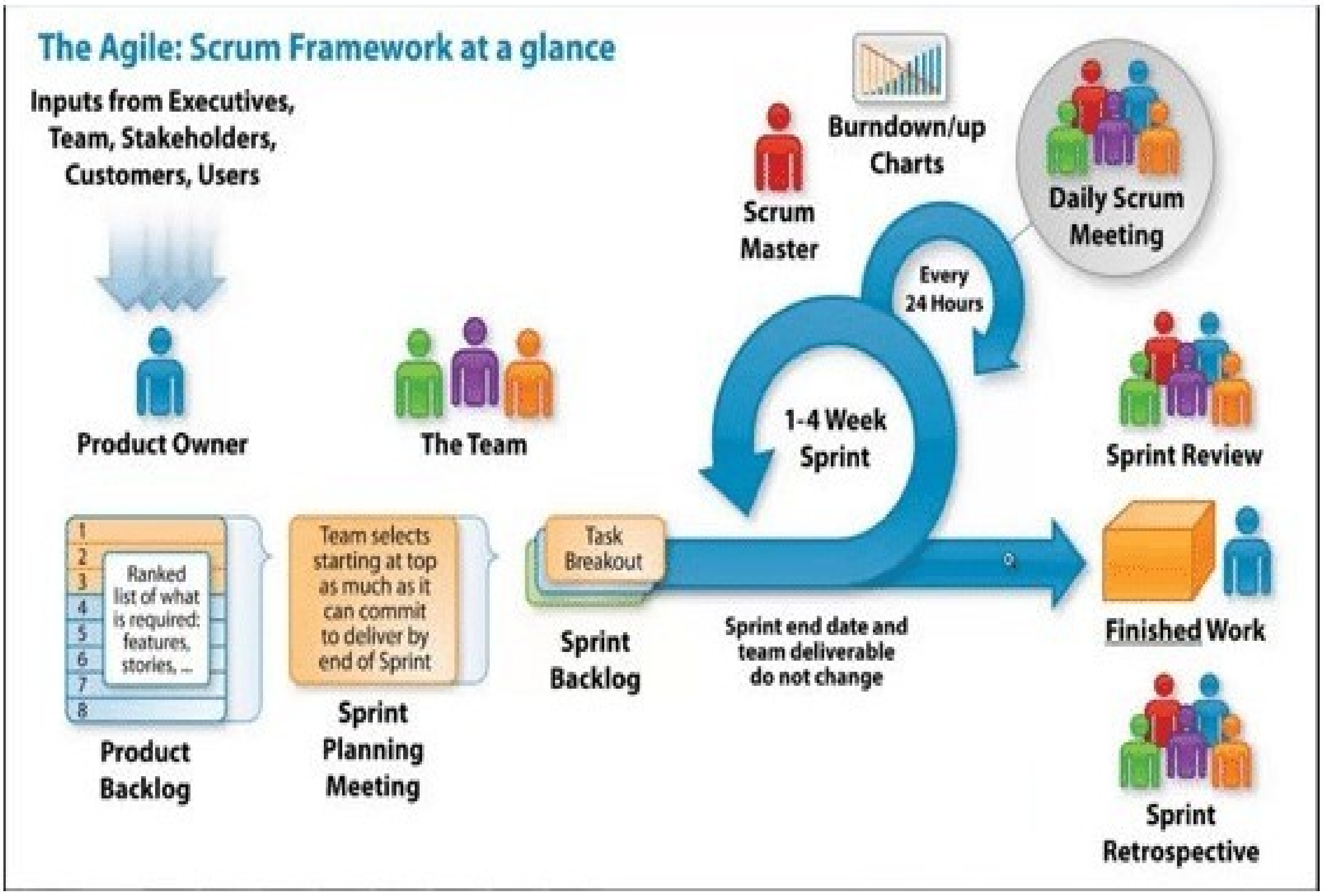
Scrum team:-

- The Product Owner
- The Scrum Master
- The Development Team

2. **Scrum Events (Ceremonies):-**
   - The Sprint
   - Sprint Planning meeting
   - The Daily Scrum Stand-up meeting
   - The Sprint Review
   - The Sprint Retrospective

3. **Scrum Artifacts :-**
   - Product Backlog
   - Sprint Backlog
   - Product Increment

4. **Scrum Rules :-**The rules of agile Scrum should be completely up to the team and governed by what works best for their processes. The best agile coaches will tell teams to start with the basic scrum events listed above and then inspect and adapt based on your team's unique needs so there is continuous improvement in the way teams work together.


The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
1
2
3
4
5
6
7
8
Ranked list of what is required: features, stories, ...
Product Backlog
Team selects starting at top as much as it can commit to deliver by end of Sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Sprint end date and team deliverable do not change
Sprint Review
Finished Work
Sprint Retrospective

STLC:-

STLC stands for Software Testing Life Cycle. STLC is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. It is the sequence of activities carried out by the testing team from the beginning of the project till the end of the project. A good software tester is expected to have good knowledge of the STLC lifecycle and its activities.

- STLC is an integral part of Software Development Life Cycle (SDLC). But, STLC deals only with the testing phases.
- STLC starts as soon as requirements are defined or SRD/SRS (Software Requirement Document / Software Requirement Specification) is shared by stakeholders.
- STLC provides a step-by-step process to ensure quality software.
- In the early stage of STLC, while the software or the product is developing, the tester can analyze and define the scope of testing, entry and exit criteria and also the Test Cases. It helps to reduce the test cycle time along with better quality.
- As soon as the development phase is over, the testers are ready with test cases and start with execution. This helps to find bugs in the initial phase.

There 5 phases is STLC and they are:-

1. **TEST PLANNING AND CONTROL**
2. **TEST ANALYSIS AND DESIGN**
3. **TEST IMPLEMENTATION AND EXECUTION**
4. **EVALUATION OF EXIT CRITERIA AND REPORTING**
5. **TEST CLOSURE ACTIVITY**

1. TEST PLANNING AND CONTROL:-

a. Test Planning:-

Test Planning phase starts soon after the completion of the Requirement Analysis phase. Test Planning is an activity of defining the objectives of testing and the specification of test activities in order to meet the objectives and mission. During test planning, we make sure that we understand goals of customer, stakeholders and project. We also analyse the risks which testing is intended to address. The main task of planning is to determine the test strategy or test approach. Test planning is an activity which is done by Sr. Tester/Test Manager/Test Lead/Test Coaches etc..Here the Sr. Tester is going to create a document called as “Test Plan document”. It is created with help of a standard template i.e. (IEEE829 Test plan standard template)- will described in detail in 4th chapter.

Test planning major tasks are:

- Determine the scope and risks and identify the objective of testing.
- Determine the test approach.
- Implement the test policy and/or test strategy.
- Determine the required test resources
- Schedule test analysis and design tasks, test implementation, execution and evaluation.
- Determine the exit criteria.

b. Test Control:-

Test Control is the ongoing activity of comparing actual progress against the plan. We need to control and measure progress against the plan. We need to control and measure progress against the plan to compare actual progress against the planned progress, and we should report to the project manager and customer on the current status of testing, including any changes or deviations from the plan. We'll need to take actions where necessary to meet the objectives of the project. Such actions may entail changing our original plan, which often happens.

Test control has following major tasks:

- Measure and analyse the results of reviews and testing
- Monitor and document progress, test coverage and exit criteria
- Provide Information on Testing
- Initiate corrective actions
- Make decisions

Deliverables (Outcome) of Test Planning and control phase are:-

- Test Plan document created by Sr.Tester/TL/TM
- Test Strategy document
- Best suited Testing Approach

2. TEST ANALYSIS AND DESIGN:-

a. Test Analysis:-

Test Analysis is process of analyzing all the documents from which the requirements of a component or system can be inferred and defining test objectives. It covers "**WHAT**" is to be tested in the form of test conditions and can start as soon as the basis for testing is established for each test level.

b. Test Design:-

Test design is a process of defining the test conditions and creating the Test Scenario document. Test analysis and design is done by Sr. Tester/TM/TL

Deliverables (Outcome) of Test analysis and design phase are:-

- Test Scenario document created by Sr. Tester/TM/TL

-

3. TEST IMPLEMENTATION AND EXECUTION:-

a. Test Implementation:-

Test Implementation is an item or event of a component or system that could be verified by one or more test cases (ex: function, transaction, feature, etc.).It Covers "**HOW"** something is to be tested by identifying test cases with step wise elaboration for the test conditions (from Test Analysis and design phase).This phase can start for a given Test Level once Test Conditions are identified and enough information is available to enable the production of Test Cases. In other words, a test case is a set of input values, execution preconditions, expected results and execution post-conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement.

This Phase is done by Sr. Tester/TL/TM (i.e. test case is developed by Sr.Tester)

b. Test Execution:-

Test execution is the process of executing the software and comparing the expected and actual results. This Phase is done Jr. Tester. Here the Jr. Tester executes the test case document and finds the defect from the Software(build) and they creates documents such as Defect Report and Execution Log.

The following points need to be considered for Test Execution.

- In this phase, the testing team performs actual validation of Software based on prepared test cases and compares the stepwise result with the expected result.
- The entry criteria of this phase is completion of the Test Plan and the Test Cases Development phase, the test data should also be ready.

- The validation of Test Environment setup is always recommended through smoke testing before officially entering the test execution.
- The exit criteria requires the successful validation of all Test Cases; Defects should be closed or deferred; test case execution and defect summary report should be ready.

Activities for Test Execution:-

The objective of this phase is real time validation of s/w before moving on to production/release. To sign off from this phase, the QA team performs different types of testing to ensure the quality of product. Along with this defect reporting and retesting is also crucial activity in this phase.

Deliverables (Outcome) of Test Implementation and Execution phase are:-

- Test Case document created by Sr. Tester.
- Defect Report created by Jr. Tester
- Execution Log created by Jr. Tester

4. EVALUATION OF EXIT CRITERIA AND REPORTING:-

a. Evaluation of exit criteria:-

Exit criteria is set of agreed conditions with stakeholders based on which you can officially mark the testing process to be completed for a particular test level. Exit criteria should be set for each test level. In exit criteria evaluation we assess the test execution against the defined and agreed exit criteria for a particular test level. Based on this evaluation we can decide if we have actually done enough testing for that level to mark it officially complete.

Some common points that are mostly present in exit criteria are as follows:

- All critical, high and medium priority test cases are executed.
- No blocker, critical or high priority defects are outstanding.
- All medium and low priority defects are triaged and agreed to be differed or fixed.

- Agreed fixes for medium and low priority defects completed and retested.
- All the test documentation like test plan, test cases are documented and up to date in the test management tool.

<u>The main tasks for evaluating exit criteria are as follows:</u>

- Checking the test logs against the defined exit criteria. For example, checking the test execution progress for all the critical, major and medium test cases. Checking the status of all outstanding defects and how many needs to be fixed and retested to fulfill the exit criteria.
- Secondly we need to assess if there are more tests that need to be executed because of the product quality concerns (i.e. you found more than expected integration issues). You might want to add more tests for execution to lower the quality risks specified in the the exit criteria. Sometimes you might also need to modify the exit criteria with agreement from stakeholders if it had very strict criteria set initially but at current stage of project those criteria pose minimum risk so that exit criteria is met.

b. <u>Reporting:-</u>

Here we write the test summary report .<u>Test summary report contains the summarized description of all the activities of testing process and it mention details about the defect and testing status of the process.</u>
When you are evaluating the exit criteria it's not only Development team and Test team need to know the outcome but also the broader set of stakeholders need to know what happened in testing levels so that they can make decisions about the software. For example, is it good to start the next testing level or if all the test levels have completed can the software go live in production.

<u>Deliverables (Outcome) of Evaluation of Exit Criteria and Reporting phase are:-</u>

- Test Summary report

5. <u>TEST CLOSURE ACTIVITY:-</u>

Test closure activities are those activities which are performed at the end of the testing process. These are usually performed after the product is delivered. After determining that test execution has been complete the data from completed test activities need to be collected and consolidated. You need to analyze the data to find out facts and numbers about the testing activities during project cycle. Test closure activities are done mostly after the product is delivered but there are other instances as well where test closure is done like, if project got cancelled or after maintenance release is done. Here in this phase, all the documents which were created throughout the project cycle is collected and consolidated together and it is stored in the database. So that if in future if there is any similar projects comes up then we can reuse the test documents for that similar project. This will make sure that mistakes should not be repeated which happened before . This phase is also done for knowledge sharing and the learning purpose

Methods of Testing:-

1. **Black Box Testing:-**The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

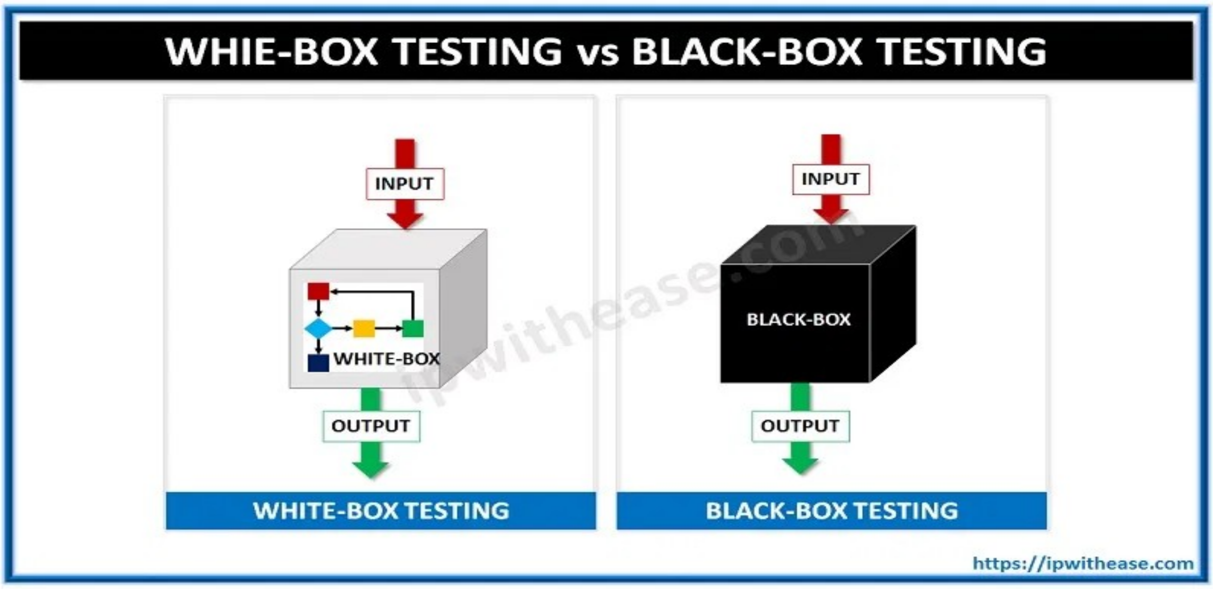
   Black-Box Testing Techniques:-

- Equivalence Class partitioning.
- Boundary value analysis.
- Error Guessing.

2. **White Box Testing:-** White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

   White-Box Testing Techniques:-

- Statement coverage.
- Decision coverage.
- Condition Coverage
- Path Coverage
- Control testing
- Data Flow testing etc..



Differences between Black Box Testing vs White Box Testing:

<u>Black Box Testing</u>	**<u>White Box Testing</u>**
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
It is functional test of the software.	It is structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after detail design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed or Functional testing.	It is also called as Clear box , Glass box , Transparent testing.

Black Box Testing	**White Box Testing**
It is least time consuming.	It is most time consuming.
Example: search something on google by using keywords	**Example:** by input to check and verify loops

Black-Box Testing Techniques:-

- Equivalence Class partitioning:- Group the condition into Equal Class Partition and select any random value from each partition. (Note:- there are 2 classes, they are 1.Valid Class 2.Invalid Class)
- Boundary value analysis:- Group the condition into Equal Class Partition and select Maximum and Minimum value from from the edges of valid class partition or +1 and -1 to the edges of valid class partition.
- Error Guessing:- It is an Experienced based techniques where testing of software is done based on the testers skills and knowledge and experience

CHAPTER 2
VERIFICATION, VALIDATION, and V-Model

VERIFICATION	**VALIDATION**
1) It is a QA Activity	1) It is a QC Activity
2) Are we Building the Product right?	2) Are we Building the right Product?
3) Verification is the process of evaluating the work products of a software development lifecycle to check if we are in the right track of creating the final product.	3) Validation is the process of evaluating the final product to check whether the software meets the business needs or fulfils customers' needs.
4)) It is done by reading the documents manually	4) It is done by executing the software
5) It is also called as Static Testing	5) It is also called as Dynamic testing
6) It does ***not*** involve executing the code	6) It always involves executing the code
7) It is defect Preventive Approach	7) It is a defect detective and corrective approach
8) Here, the root cause(location) of the defect is found	8) Here, the root cause(location) of the defect is not found
9) The verifying process includes checking documents, design, code, and program	9) It is a dynamic mechanism of testing and validating the actual product
10) The Cost of Fixing the defect during verification is cheaper as compared to the cost of defect fixing during validation	10) The Cost of Fixing the defect during validation is costly as compared to the cost of defect fixing during verification

11) Verification happens at all phases of SDLC	11) Validation happens only at the last 3 phases of SDLC
12) Verification Techniques are:- 1) Walkthrough 2) Inspection 3) Review	12) Validation Techniques are:- (Levels of Testing) 1) Unit Testing 2) Integration Testing 3) System Testing 4) Acceptance Testing
13) It is done by the seniors of the respective phases	13) It is mostly done by the Testers

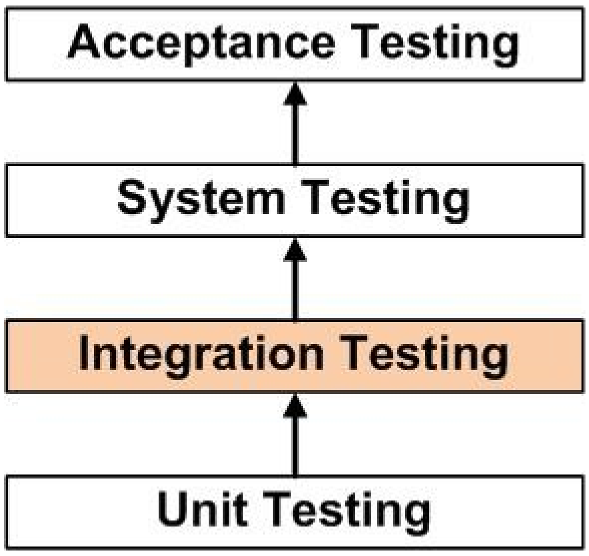
Verification Techniques:-

1) **Walkthrough:**
   - It is an informal Process because here, Rules and Regulations is not followed, No seniors are involved , No proper Documentations that’s why it is called as Informal Process
   - It happens between the friends or colleagues.
   - Walkthrough is just a casual talk between two or more colleagues were the colleague give their feedback or share the view-point regarding the work product to the author.
   - Author leads the Walkthrough.
2) **Inspection :-**
   - It is Semi-formal process because here, Proper Rules and Regulation are followed, with Proper documentation, Seniors are involved but here the final Decision is not made that’s why it is called Semi-formal process.
   - There are 5 people who are involved in Inspection:-
     - ➢ Author:- the person who created the documents and whose documents is under inspection
     - ➢ Moderator:- the person who decides and inform all the people about the date, time and venue for the inspection.
     - ➢ Inspector/Reviewer:- the person who does the inspection of the product. They are the domain experts.

- Reader:- the person who reads the documents for everyone in the inspection meeting
- Recorder:- the person who note down all the events which occurred during the meeting, along with the defects details , corrections, and follow-up details. In short. he/she makes a note of Minutes of meeting(MOM)

3) **Review:-**
   - It is a formal Process because here the final decision is been taken that "whether the phase is complete and can we move to the next phase" and this decision is taken based on the correctness and completeness of the document.
   - It is done by the senior most person (managers) of the project team.

Validation Techniques:-



1) **Unit Testing:-**
   - Testing a small piece of executable code which gives the desirable output is called as Unit testing.
   - It is done by the developers (White Box method is used for doing the unit testing). Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying every unit of the code used to implement functional requirements.
   - "Unit testing is the method of verifying the smallest piece of testable code against its purpose." If the purpose or requirement failed then the unit test has failed.
   - In simple words, it means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

- **UNIT TESTING** is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class.

2) <u>**Integration Testing:-**</u>

- It is done to test the communication link / Interfaces between two or more modules.
- When two or more modules are ready then integration testing is done on it.
- (What is modules:- collections of several units is called as one modules).
- **INTEGRATION TESTING** is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. For eg:- During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. For example, whether the cap fits into the body or not.
- **When is Integration Testing performed?:-** Integration Testing is the second <u>level of testing</u> performed after <u>Unit Testing</u> and before <u>System Testing</u>.
- **Who performs Integration Testing?:-**Developers themselves or independent testers perform Integration Testing.
- Point to remember before doing integration testing:-Ensure that you have a proper Detail Design document where interactions between each unit are clearly defined. In fact, you will not be able to perform Integration Testing without this information. Make sure that each unit is unit tested before you start Integration Testing.
- **There 3 approaches of doing Integration Testing:-**
    i. <u>**Top-Down Approach:-**</u>
        1. When the parent module is ready and child module is not ready then Developers create a dummy code for the child module called as "**Test Stub**" for doing the integration testing.

2. *Top Down* is an approach to Integration Testing where top-level units are tested first and lower level units are tested step by step after that. This approach is taken when top-down development approach is followed. Test Stubs are needed to simulate lower level units which may not be available during the initial phases.

ii. **Bottom-Up Approach:-**

1. When the parent module is not ready and child module is ready then Developers create a dummy code for the Parent module called as "**Test Driver**" for doing the integration testing.
2. *Bottom Up* is an approach to Integration Testing where bottom level units are tested first and upper-level units step by step after that. This approach is taken when bottom-up development approach is followed. Test Drivers are needed to simulate higher level units which may not be available during the initial phases.

iii. **Big Bang Approach:-**

1. *Big Bang* is an approach to Integration Testing where all or most of the units are combined together and tested at one go. This approach is taken when the testing team receives the entire software in a bundle. So what is the difference between Big Bang Integration Testing and System Testing? Well, the former tests only the interactions between the units while the latter tests the entire system.
2. This Approach always leaves a doubt and consumes lots of time. That's y this approach is not used mostly for integration testing.

3) **System Testing:-**

- SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
- The process of testing an integrated system to verify that it meets specified requirements.
- Here the software is tested with the scope of release whether the software is stable and completely eligible for releasing it to the client or not.
- **When is it performed?:-** System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing.
- **Who performs it?:-** Normally, independent Testers perform System Testing.

- In here both types of testing is carry forward – Functional Testing as well as Non-Functional Testing
- Testers use Black-Box Method for doing System testing.
- **System Testing** is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

4) **Acceptance Testing:-**

- ACCEPTANCE TESTING is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
- Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. For eg:- During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed. Once System Testing is complete, Acceptance Testing is performed so as to confirm that the ballpoint pen is ready to be made available to the end-users.
- Acceptance Testing is done by the Customer or client or client representative.
- Here the customer takes the decision whether to accept the software or not.
- There are two types Acceptance Testing:-

  i. **Alpha Testing:-**
     1. It is also called as Internal Testing / Factory testing.
     2. Here the users or customers come at the development environment(developer's site) and does the testing in presence of the development teams(programmer and testers).
     3. The identified bugs are resolved quickly by the development team there itself.

  ii. **Beta Testing:-**
     1. It is also called as External testing / Field testing.

2. Beta testing also known as user testing takes place at the Client environment(end users site) by the end users to validate the usability, functionality, compatibility, and reliability testing.
3. Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product.
4. In Case , if any defect(failure) found during the beta testing. The identified defect(failure) are reported to the development team for defect fixing.

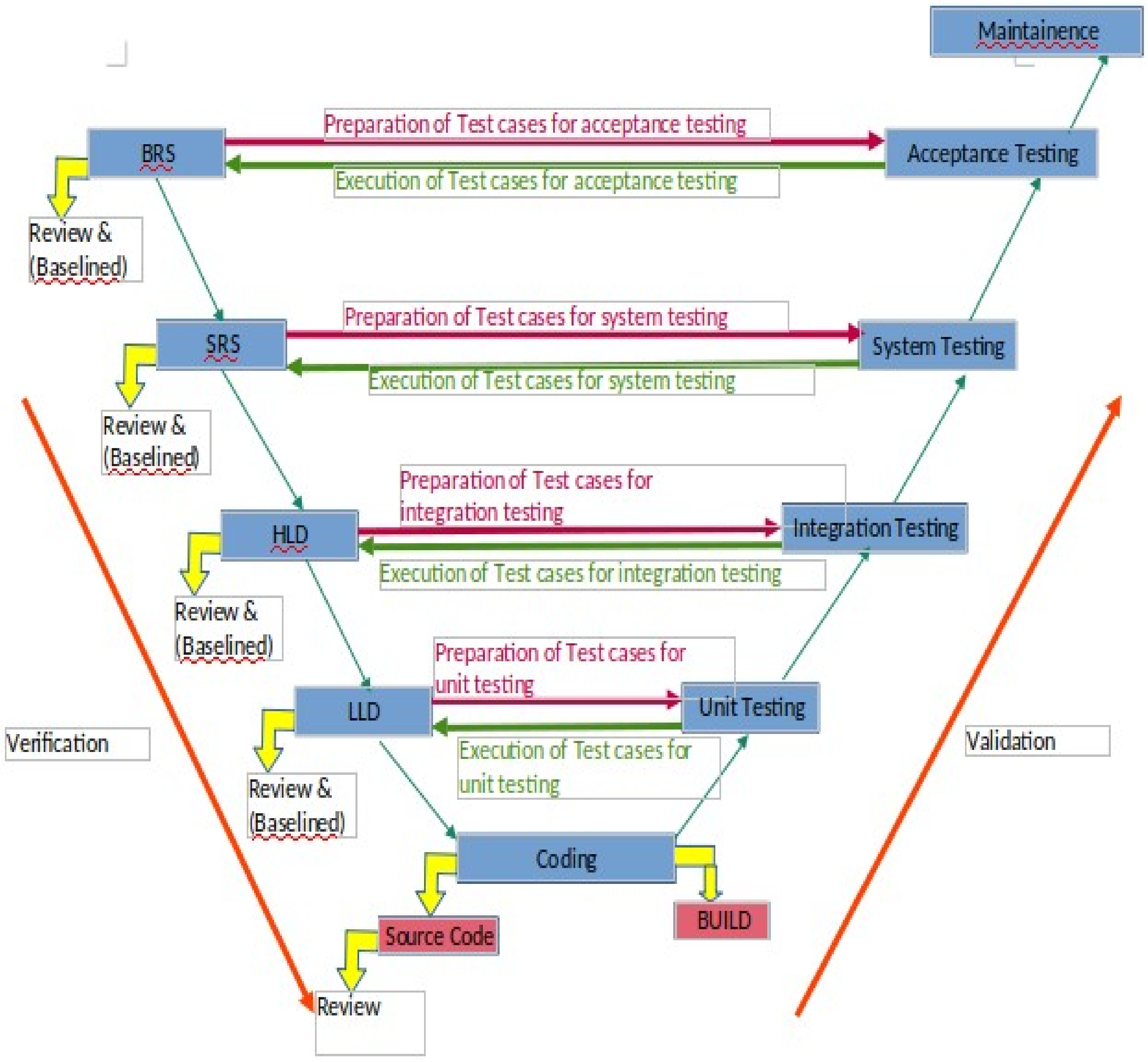
V-Model :-

When to use V Model?

V model is applicable when:

- Requirement is well defined and not ambiguous.
- Acceptance criteria are well defined.
- Project is short to medium in size.
- Technology and tools used are not dynamic.

PROS(Advantages)	CONS(Disadvantages)
● Development and progress is very organized and systematic ● Works well for smaller to medium sized projects. ● Testing starts from beginning so ambiguities are identified from the beginning. ● Easy to manage as each phase has well defined objectives and goals. ● Simple and easy to use. ● Avoids the downward flow of the defects. ● Proactive defect tracking – that is defects are found at early stage.	● Not suitable for bigger and complex projects ● Not suitable if the requirements are not consistent. ● No working software is produced in the intermediate stage. ● No provision for doing risk analysis so uncertainty and risks are there. ● Very rigid and least flexible. ● Software is developed during the implementation phase, so no early prototypes of the software are produced.


Maintainence
Preparation of Test cases for acceptance testing
BRS
Acceptance Testing
Execution of Test cases for acceptance testing
Review & (Baselined)
Preparation of Test cases for system testing
SRS
System Testing
Execution of Test cases for system testing
Review & (Baselined)
Preparation of Test cases for integration testing
HLD
Integration Testing
Execution of Test cases for integration testing
Review & (Baselined)
Preparation of Test cases for unit testing
LLD
Unit Testing
Execution of Test cases for unit testing
Verification
Validation
Review & (Baselined)
Coding
Source Code
BUILD
Review

V-Model

CHAPTER 3

TYPES OF TESTING

1. Retesting
2. Regression testing
3. Smoke testing
4. Sanity testing
5. Security testing
6. Performance testing
    a. Load testing
    b. Stress testing
    c. Endurance/Soak testing
    d. Spike testing
    e. Volume testing
7. Installation testing
8. Re-installation testing
9. Un-installation testing
10. Recovery testing
11. Configuration testing
12. Compatibility testing
13. Usability testing
14. User interface testing
15. Localization testing
16. Internationalization testing

1) Functional Testing:-

1. **FUNCTIONAL TESTING** is a type of software testing whereby the system is tested against the functional requirements/specifications.
2. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application.
3. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions.

4. During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester.
5. Functional testing is normally performed during the levels of System Testing and Acceptance Testing.
6. Typically, functional testing involves the following steps:
   - Identify functions that the software is expected to perform.
   - Create input data based on the function's specifications.
   - Determine the output based on the function's specifications.
   - Execute the test case.
   - Compare the actual and expected outputs.

2) <u>Retesting:-</u>

1. During the **test execution** period, when tester executes the test cases then if there is a difference between the expected and actual result **i.e. defect** , that defect is reported to the developer(Programmer) by the tester for defect fixing. Once after the developer completes the defect fixing then they report back the updated build to the tester. Here the tester then re-executes the same fail test case in which the defect was found **<u>to confirm</u>** that the defect is been removed or not. This process of re-executing the fail test case with the same input data is called **<u>Retesting. It is also called as Confirmation testing</u>**
2. **<u>What is Retesting? :-</u>** Re-execution of failed test case is called as Retesting.
3. **<u>When to do Retesting? :</u>**- After the defect fixing by developer.
4. **<u>Why there is need of doing Retesting?</u>** :- To confirm whether the developer has removed the defect from s/w or not.
5. **<u>Who?</u>** :-Software tester
6. **<u>How?</u>** :- By executing the fail test cases with the same input data

3) <u>Regression Testing:-</u>

1. Regression Testing is done to check the impact of changed part on the unchanged part whether the changes has produced any new defect or not.
2. Regression testing is done when there is a changes such as :-
   a. Defect fixing

   b. Any new Requirement/features is been added.
   c. Modifying the existing requirement/feature
   d. Removing the existing requirement/feature
3. Regression testing is repetitive in nature that's why it is mostly done through automation testing using automation tools such as Selenium, QTP(Quick Test Professionals), RFT(Rational Functional Tester).
4. **What is Regression testing? :-** Re-execution of passed test case is called as Regression testing.
5. **When to do Regression testing? :**- After if there is any changes made/implemented in the s/w.
6. **Why there is need of doing Regression testing?** :- To check whether the changes made in the s/w is not producing any new defect or not
7. **Who?** :-Software tester
8. **How?** :- By executing the pass test cases which is found by doing **Impact Analysis** on **RTM**
9. **What is Impact Analysis:-**
   a. It is the process of identifying the directly linked passed test cases which is related with the failed test cases or changes done , from RTM.
   b. This process gives us the answer of "How much regression need to be done?" i.e. find out the area which are impacted due to the changes in the s/w.
   c. Impact analysis is basically analyzing the impact of the changes in the deployed application or product.
   d. It tells us about the parts of the system that may be unintentionally affected because of the change in the application and therefore need careful regression testing.
10. **What is RTM:-**
    a. It is a document which shows the cross reference relationship between the Requirement and Test cases.
    b. It is also called as "**Bi-directional Traceability Matrix**"
    c. **Requirement Traceability Matrix (RTM**) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle.

d. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

4) <u>**Smoke testing:-**</u>

1. **Checking the all basic Functionality of the s/w to verify whether it is stable and eligible or not for further testing process is called as SMOKE TESTING**.
2. Smoke testing is a type of software testing and the purpose of Smoke Tests it to confirm whether the Testing team can proceed with further testing. Smoke tests are a minimal set of tests run on each build.
3. Smoke testing is a process where the software build is verified before it is deployed to Testing environment to ensure the stability of the application. That's why it is called as **"<u>Build verification Testing</u>"** or ***"Confidence Testing"*** .After verifying the build testers accepts the build that's why it is also called as **"<u>Build Acceptance Testing</u>". <u>It is also called as "Shallow and Wide Testing"</u>**
4. In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing.
5. It is done by just with mouse clicks without going deeper and without giving keyboard inputs. (But in rare cases, there will be a need of keyboard inputs)

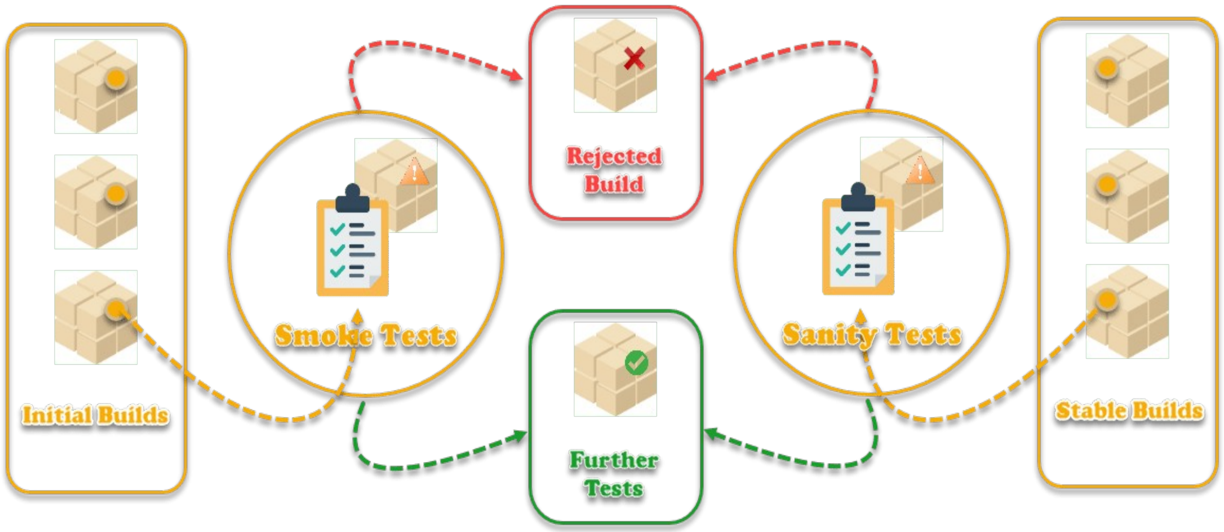
5) <u>**Sanity testing:-**</u>

1. **Checking the basic Functionality of the s/w after minor changes is done in the s/w is called as SANITY TESTING.**
2. **It is mainly done during the pre-release phase where the software build is going to hit the production.**
3. **It is done on the matured builds whereas smoke is done on the initial (fresh) builds. It is sub-set of Regression testing(<u>Sub-Regression testing</u>). It is also called as <u>"Narrow and Deep Testing"</u>**
4. Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are

introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

5. It is done by giving keyboard inputs.
6. **According o ISTQB**, SMOKE and SANITY are same because in both we talk about the software build, we talk about the functionality, we talk about the rejection if build's if builds health is not well for further feasible testing.

Difference between Smoke and Sanity testing



Both the terms Smoke and Sanity are used interchangeably to a great extent in the IT industry, but there are also a few critical differences between them, as explained below:

Feature	*Smoke Testing*	*Sanity Testing*
Test Coverage	Smoke testing is a shallow and broad approach.	Sanity testing is usually a narrow and in-depth approach.
Motive	It is designated to touch every part of the application quickly.	It basically emphasis on the small sections of the application and check whether it is working correctly after a minor change.
Technique	The test cases for smoke testing can be either manual or automated or sometimes a hybrid approach.	A sanity test is performed generally without test scripts or test cases but manually.
Performed By	The application developers or QA team or testing team perform this testing.	The testing team or QA team usually performs sanity.
Subset	It can be a subset of acceptance testing or regression testing.	Sanity testing is more of a Monkey and Exploratory testing. It is also called as subset of regression testing
Documentation	Properly documented. Smoke testing is usually managed separately as a Smoke Test Suite.	No proper documentation is available. Sanity testing is generally performed with the experience only and focus on the defect and nearby area.

Importance in the Software Industry:

Similarities	*Explanation*
Saves Time	Smoke and Sanity tests are efforts to save time by quickly determining if an application is working correctly or not. Also, it ensures that the compilation is eligible for rigorous testing.
Save cost	The saving of time and effort leads to saving the cost of testing the application. Employing the correct approach and eliminating the mistakes in the early stage, lowers the effort and time, thus minimizing the damage.
Integration risk	We perform end-to-end tests on each build so that functionality-based problems get discovered earlier. So, the risk of having integration issues minimizes.

Quality improvement	Here, the main problems are detected and corrected much earlier in the software test cycle, which increases the quality of the software.
Evaluation of progress	It is easier for project managers to assess the progress of development. Since with each compilation, we certify that the product from end to end is working correctly after the addition of new features.

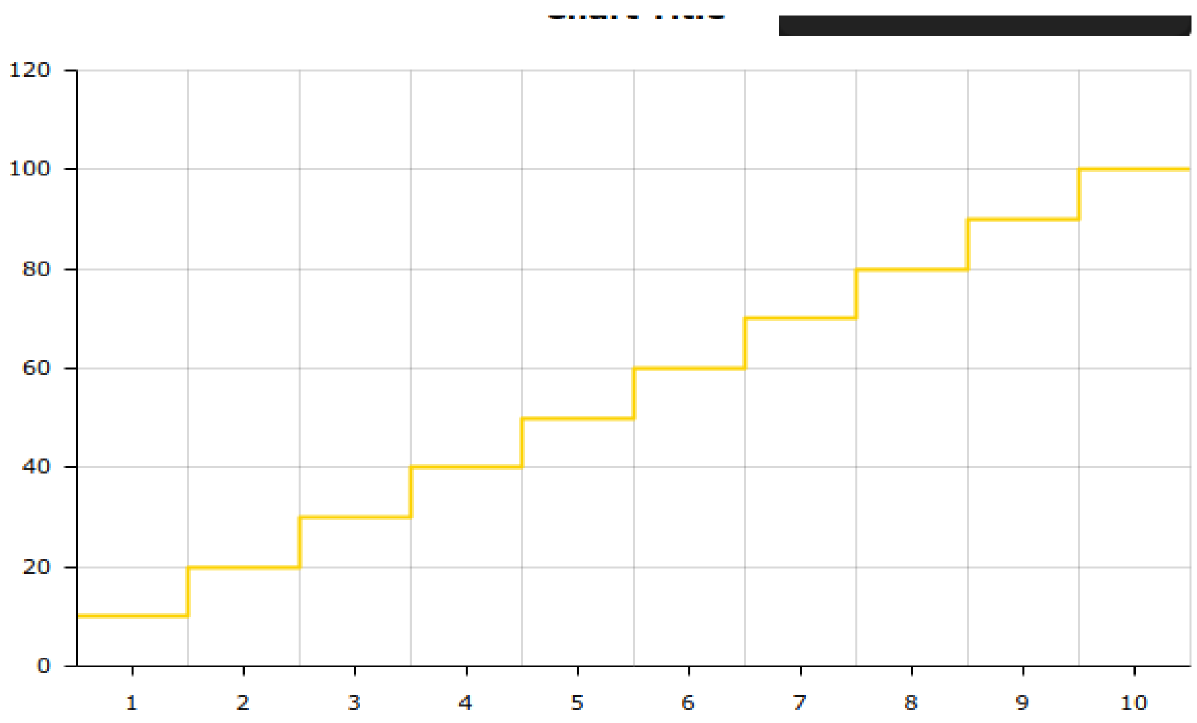
6) <u>Security testing:-</u>

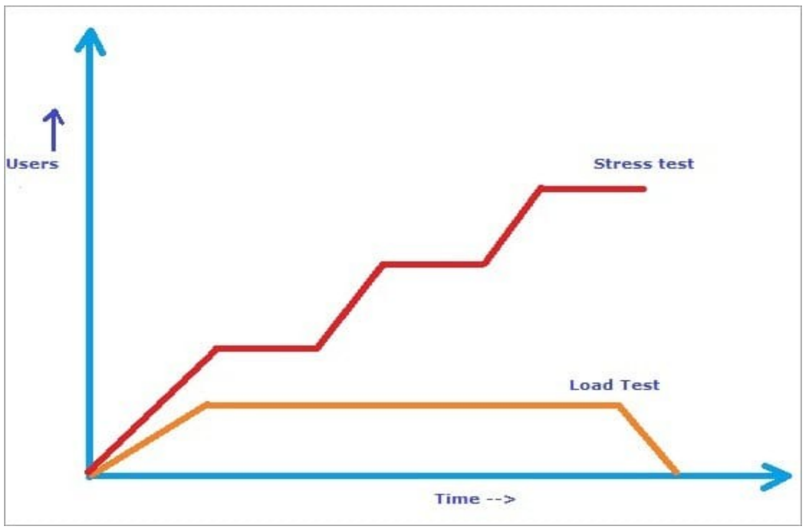
- It is a type of non-functional testing.
- It is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders.
- Security testing is basically a type of software testing that's done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- It is a process to determine that an information system protects data and maintains functionality as intended.
- The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software's and hardware's and firewall etc.
- There are several measures of testing carried out for doing the security testing. And they are as follows:-
  - **Authentication:** Only the valid user should be able get access to the system
  - **Authorization:** Only the authenticated user should be able to access to controls as per the user's role
  - **Vulnerability Scanning:** This is done through automated software to scan a system against known vulnerability signatures.
  - **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.

- **Penetration testing:** This kind of testing simulates an attack from a malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
- **Risk Assessment:** This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.
- **Security Auditing:** This is an internal inspection of Applications and Operating systems for security flaws. An audit can also be done via line by line inspection of code
- **Ethical hacking:** It's hacking an Organization Software systems. Unlike malicious hackers, who steal for their own gains, the intent is to expose security flaws in the system.
- **Session control:** user should be log out automatically from application after particular required session period

7) Performance testing:-

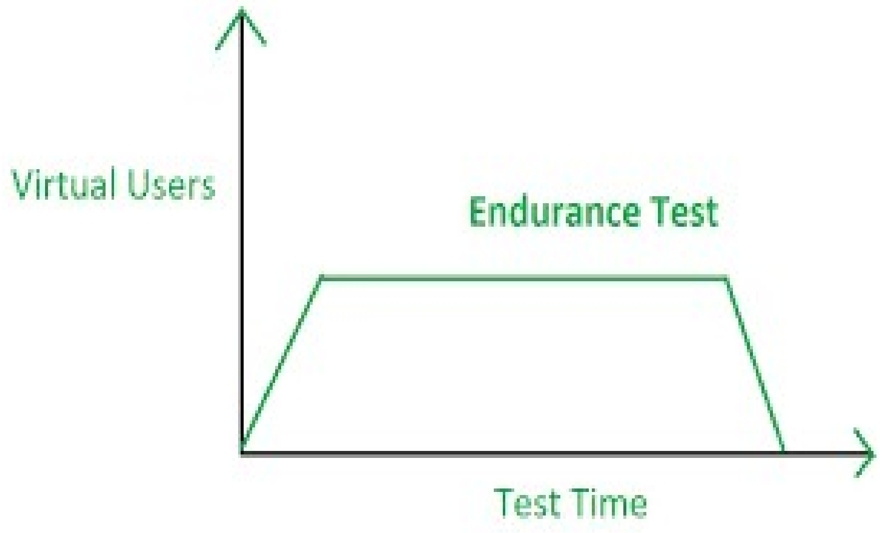
- It is type of Non- Functional Testing.
- To check the behaviour of the application or software with respect to the **response time** is called as Performance Testing. **(Response time**:- the time duration between the first request and last response is called as Response time)
- **There are several type of Performance Testing:-**
  ➔ **Load testing:-**Testing the application with the peak load is called as Load testing.**(Peak load:-** total No. Of users given by customer**)**.It is conducted to understand the behaviour of the application under a specific expected load.

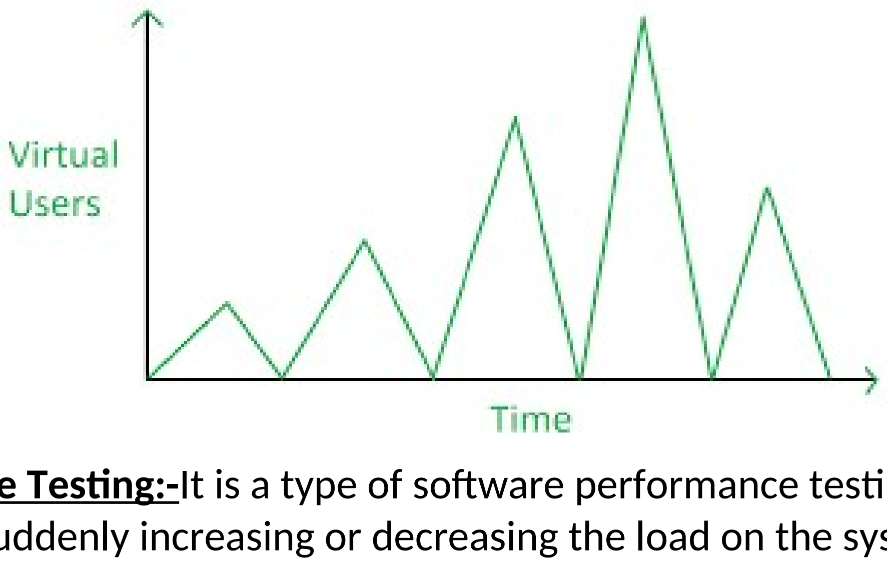




➔**Stress Testing:-**Testing the application with beyond the peak load is called stress testing. The main purpose of stress testing is to determine the **Degradation point** (Saturation point) and **Crash point** (Breaking point) of the application when the application is loaded with beyond the peak load.

➔**Endurance Testing:-**Testing the application with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. It is also known as **Soak testing**. The main purpose of the Endurance testing is to check the memory leaks. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly. The main goal is to discover how the system behaves under sustained use. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.





- **<u>Spike Testing:-</u>**It is a type of software performance testing that is done by suddenly increasing or decreasing the load on the system or software application. The goal of spike testing is to determine whether the system will fail or survive in case of dramatic changes in load. The objective of Spike Testing is: To evaluate the behaviour of the system or software application under the load changed abruptly And To observe the performance of the system under abruptly changed load.

➔ **Volume Testing:-** Testing the application with respect to the database size is called as volume testing. Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application's performance on it. The purpose of **volume testing** is to determine system performance with increasing volumes of data in the database.

8)

CHAPTER 4
TEST PLANNING

1) WHAT IS TEST PLAN:-

Software test plan document is a guide book for testing process. It is required for the successful execution of testing process for a project. It contains comprehensive information to carry out the testing activities. A **TEST PLAN** is a document describing software testing scope and activities. It is the basis for formally testing any software/product in a project.
A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

2) WHO Creates the TEST PLAN document:-

Senior Tester/ Test Manager / Test Lead/ Test Co-ordinator etc….

3) WHY THERE IS NEED OF TEST PLAN:-

- To get an effective testing process with proper/good outcome
- To Avoid the random testing.
- To make there is not any missing features
- To Optimizes the utilization of resources
- To have a good Risk management

What are the Benefits of Test Plan?

Value of writing a test plan is tremendous. It offers a lot of benefits like:

- It serves as a roadmap to the testing process to ensure your testing project is successful and helps you control risk.
- Proper test plan provides a schedule for testing activities, so, you can have a rough estimate of time and effort needed to complete the software project
- It clearly defines the roles and responsibilities of every team member, outlines the resource requirements which are essential to carry out the testing process
- Planning and a test plan encourages better communication with other project team members, testers, peers, managers, and other stakeholders

- Helps people outside the test team such as developers, business managers, customers properly understand the details of testing

So, what happens when one doesn't have a test plan?

- Scope of testing will not be properly defined
- There will be a misunderstanding about roles and responsibilities and this could lead to important tasks left undone
- Test team will not have clear test objectives, without which critical and essential system characteristics will not be adequately tested
- Test team will not be aware of when the test process ends, which will compromise the quality, functionality, effectiveness, and efficiency of the software

4) <u>WHEN TO CREATE A TEST PLAN:-</u>

As soon as **<u>SRS</u>** document is base-lined in the Requirement Gathering and Analysis phase and when Development Plan, Project Plan are ready then the test plan document is created.

5) <u>HOW TO CREATE THE TEST PLAN:-</u>

Test Plan Template

IEEE (Institute of Electrical and Electronics Engineering), an international institution that defines standards and template documents which are globally recognized. It has defined the IEEE 829 standard for system and software documentation. This IEEE 829 standard Test Plan Template specifies the format of creating the test plan. There are 17 attributes in the standard test plan template. They are as follows as:-

The table below lists out the test plan parameters according to the IEEE 829 standard test plan template.

<u>Parameter</u>	<u>Description</u>
Test Plan Identifier	Uniquely identifies the test plan & may include the version number
Introduction	Sets objectives, scope, goals, resource & budget constraints
Test Items	Lists the systems and subsystems which are to be tested

Features to be Tested	All the features & functionalities to be tested are listed here
Features not to be Tested	Lists the features of the software product that need not be tested
Approach	Has sources of test data, inputs and outputs, testing priorities
Item pass/fail	Describes a success criteria for evaluating the test results
Suspension Criteria	Has criteria that may result in suspending testing activities
Test Deliverables	Includes test cases, sample data, test report, issue log
Testing Tasks	Describes dependencies between tasks & resources needed
Environmental Needs	Lists software, hardware or other testing requirements
Responsibilities	Lists roles and responsibilities assigned to the testing team
Staffing Needs	Describes the additional training needs for the staff
Schedule	Details on when the testing activities will take place are listed
Risks	Lists overall risk of the project as it pertains to testing
Approvals	Contains signature of approval from stakeholders

The format and content of a software test plan vary depending on the processes, standards, and test management tools being implemented from company to company. Nevertheless, the following format, which is based on IEEE standard for software test documentation, provides a summary of what a test plan can/should contain.

- **Test Plan Identifier:** Provide a unique identifier for the document. (Adhere to the Configuration Management System if you have one.)
  syntax:- Project Name_project version_level of plan_plan versions
  for e.g.:- Whatsapp 12.1 MTP 1.2 or Flipkart 34.9 STP 2.0

  Test Plan Types

  There are 2 types of test plans:

  - **Master Test Plan:** A single high-level test plan for a project/product that unifies all other test plans.
  - **System Test Plan:** Test Plans for each level of testing or for different major types of testing .
    - Unit Test Plan
    - Integration Test Plan
    - System Test Plan
    - Acceptance Test Plan

    Testing Type Specific Test Plans like Performance Test Plan and Security Test Plan etc…

- **Introduction:**
  - Describe the project
  - Provide an overview of the test plan.
  - Specify the goals/objectives.
  - Specify any constraints.

- **References:** List the related documents which were used/referred to create the test plan document, with links to them if available, including the following:
  - **SRS**
  - Project Plan
  - Configuration Management Plan
  - Development Plan
  - Test document of similar kind of projects
  - Test document of previous version of same project
  - Quality Plan

- **Test Items:** List the test items/ **Modules** (software/products) and their versions which is going to be tested for this current release.

- **Features to be Tested:**
  - List the features of the Modules (software/product) to be tested.
  - Provide references to the Requirements and/or Design specifications of the features to be tested

- **Features Not to Be Tested:**
  - List the features of the software/product which will not be tested.
  - Specify the reasons these features won't be tested.

- **Test Approach:**

  - Mention the overall approach to testing.
  - Specify the testing levels [if it's a Master Test Plan], the testing types, and the testing methods [Manual/Automated; White Box/Black Box].

- **Test Deliverables:**

  List test deliverables, and links to them if available, including the following:
  - **Test Plan (this document itself)**
  - **Test Scenario**
  - **Test Cases**
  - **Defect Report**
  - Test Scripts
  - Test Metrics
  - Test Reports
  - Execution Log

Test Environment:

- Specify the properties of test environment: hardware, software, network etc.
- List any testing or related tools.
- For e.g.:-

Hardware :-	
RAM	2GB
Processor	Intel i3 core Processor with 3GHz
Operating System	Windows , Linux
Server	Apache Tomcat
Browsers	Google Chrome, Mozilla firefox
Software:-	
Pdf Reader	Adobe Pdf Reader
Defect Tracking Tool	Mantis
Test Management Tool	Test Link

Item Pass/Fail Criteria:

- Specify the criteria that will be used to determine whether each test item (software/product) has passed or failed testing.
- This criteria help to take the decision to stop the testing permanently
- For e.g.:-

EXIT Criteria:- (item pass/fail criteria)
1) All the Test Case should e executed (i.e 100% Test Coverage)
2) 90% Test case should be Pass and 10% Test case may be fail with low Severity and low Priority
3) Required quality of product (software) should be achieved.
4) All the Test Deliverables should be Updated and completed

Suspend and Resume Criteria:

- Specify criteria to be used to suspend the testing activity where the testing is stopped on temporary bases .
- Specify testing activities which must be redone when testing is resumed.
- For e.g.:-

Suspend Criteria	Resume Criteria
1) Show stopper defect found (Blocker defect)	1) Show stopper defect is resolved and retested
2) No Electricity	2) Electricity issue resolved
3) System crashed	3) System is repaired and started working
4) Network crashed	4) Network is available

Schedule:

- Provide a summary of the schedule, specifying key test milestones, and/or provide a link to the detailed schedule.

- For e.g.:-

Sr.No.	Task/Activities	Days(Hrs)	Date Range
1	Test plan Writing	2	21/4/2020 -22/4/2020
2	Test Scenario Writing	2	23/4/2020-27/4/2020
3	Test Case Writing	3	28/4/2020-30/4/2020
4	Test Case Execution	3	4/5/2020-6/5/2020
5	Defect Reporting	1	7/5/2020

Staffing and Training Needs:

- Specify staffing needs by role and required skills.
- Identify training that is necessary to provide those skills, if not already acquired.
- It can also be defined in a tabular /Paragraph format

Roles and Responsibilities:

List the responsibilities of each team/role/individual.

For e.g.:-

Sr.No.	Name of the Person	Designation	Task/Activities
1	Shreejith Mohan	Tester	Test plan Writing Test Scenario Writing Test Case Writing Test Case Execution Defect Reporting

Software Risks Issues:(Product Risk)

- List the risks that have been identified which are related to product /Quality of the application.
- For e.g.:-
    - Third party component not available
    - Government Regulations and policy changes
    - Major changes in the software requirements
    - Incorrect requirement in SRS document
    - Incorrect Design Specification
    - Inability to use and understand the new package/tools
    - Safety and Security related risk   etc.....

Risk and Contingencies: (Project Risk)

Here we mention the list of project related risk which impacts or effects on the testing process. And here we also Specify the mitigation plan and the contingency plan (Solution/Back Up) for each risk. For e.g.:-

- ***Risk:-***
    - System Crash
    - Electricity issues / Power failure
    - Tester left the company
    - Network connection is not available.
    - Delay in receiving BUILD.
    - Delay in training.
    - Less Time allocated for testing.

    - Managers have poor co-ordination with colleagues and has poor communication skills
- ***Contingencies:-(Solution /Back Up for each project related risks)***
    - Provide leadership training to respective managers
    - Reschedule the testing activities / Monitoring daily testing activities.
    - Call the system administrator for network, system related issues.
    - Hire a new tester for the project

Approvals:

- Specify the names and roles of all persons who must approve the plan.
- Provide space for signatures and dates. (If the document is to be printed.)
- for e.g.:-

Sr. No.	Name	Designation	Date	Sign	Remark
1	Shreejith Mohan	Test Manager			

EXTRA ATTRIBUTES:-

Assumptions and Dependencies:

List the assumptions that have been made during the preparation of this plan.
List the dependencies.

Entry Criteria:

- Specify the criteria that will be used to determine when to start with test case execution on the (software/product).

  For e.g.:-

  ENTRY Criteria:- (Pre-Conditions)

  1) Build must be available.
  2) Environment must be ready .
  3) Test case document must be completed .
  4) Test cases are ready to execute.

Estimate:

- Provide a summary of test estimates (cost or effort) and/or provide a link to the detailed estimation.

Test plan is a guidebook for the testing process and is vital to keep the entire testing process on the right track. If you have a solid plan and test plan document in place, chances are the testing process will go smoother. This brings us to the end of this article. Hope the things that you have learned here today will help you as you head out on your software testing journey.

Test Plan Guidelines

- Make the plan concise. Avoid redundancy and superfluousness. If you think you do not need a section that has been mentioned in the template above, go ahead and delete that section in your test plan.
- Be specific. For example, when you specify an operating system as a property of a test environment, mention the OS Edition/Version as well, not just the OS Name.
- Make use of lists and tables wherever possible. Avoid lengthy paragraphs.
- Have the test plan reviewed a number of times prior to baselining it or sending it for approval. The quality of your test plan speaks volumes about the quality of the testing you or your team is going to perform.
- Update the plan as and when necessary. An out-dated and unused document stinks and is worse than not having the document in the first place.

CHAPTER 6
USE CASE

There are 7 attributes in Use Case document:-

1. User id
2. Goal
3. Actors
4. Pre-condition
5. Path
6. Steps
7. Post-Condition

Path:- (Also called as flow)

Normal path:-(Success path)

- It is also called as Basic Path. The basic **flow** is the preferred sequence of actions in a **use case**, which delivers the desired result to a customer. This the normal flow through which the user(Actor) follows to check the functionality.
- The main flow of events describes a single path through the system. It specifies the interactions between the actor and the system for an ideal condition. It represents the most common way that the use case plays out successfully and contains the most common sequence of user-system interactions.

Alternate path:-(Success path)

- The **alternate flow** is a series of actions in non-random, repeatable order, different from the basic flow, which **still delivers the desired result** to a customer. Imagine that you want to find a friend on a social network.
- An alternate flow is a series of actions other than the basic flow that results in a user completing his or her goal. Often ,**It is considered to be an optional flow.** It means that the user has chosen to take an alternative path through the system.

Exceptional path:- (Failure path)

- In this exceptional path, the user knows the error (known error/expected error) and system knows how to handle it.

- An exceptional flow is any action that will cause the Actor to not to complete or achieve the desired result. Exception flows represent an undesirable path to the user. However, even though the exception flow has occurred the system should still react in a way that recovers the flow and provides some useful information to the user.

<u>Error path:-(Failure path)</u>

- In this Error path, the user doesn't knows the error (Unknown error/unexpected error) and system doesn't knows how to handle it.
- In Error path, it flow is any action that will cause the Actor to not to complete or achieve the desired result. Error flows represent an unexpected and undesirable path to the user. Here, when the error path is occurred the system doesn't know to react to that situation.

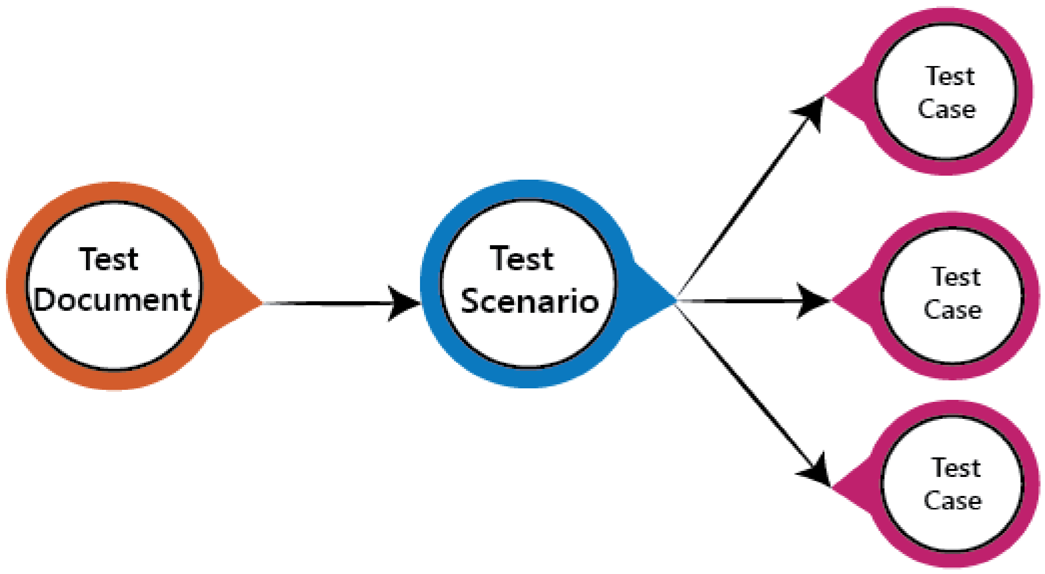
CHAPTER 7
TEST DESIGN TECHNIQUES

TEST SCENARIO:-

- The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.
- In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.
- In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.
- **Note:** *The test scenarios can never be used for the text execution process because it does not consist of navigation steps and input. These are the high-level documents that talks about all the possible combination or multiple ways or combinations of using the application and the primary purpose of the test scenarios are to understand the overall flow of the application.*
- **How to write Test Scenarios:- As a tester, follow the following steps to create Test Scenarios-**
  - ✓ Read the requirement document such as BRS (Business Requirement Specification)-if available, SRS (System Requirement Specification) and FRS (Functional Requirement Specification) of the software which is under the test.
  - ✓ Determine all technical aspects and objectives for each requirement.
  - ✓ Find all the possible ways by which the user can operate the software.
  - ✓ Ascertain all the possible scenario (Test Conditions)due to which system can be misused and also detect the users who can be hackers.
  - ✓ After reading the requirement document and completion of the scheduled analysis make a list of various test scenarios / Possibility / Multiple ways (Test Conditions) to verify each function of the software.

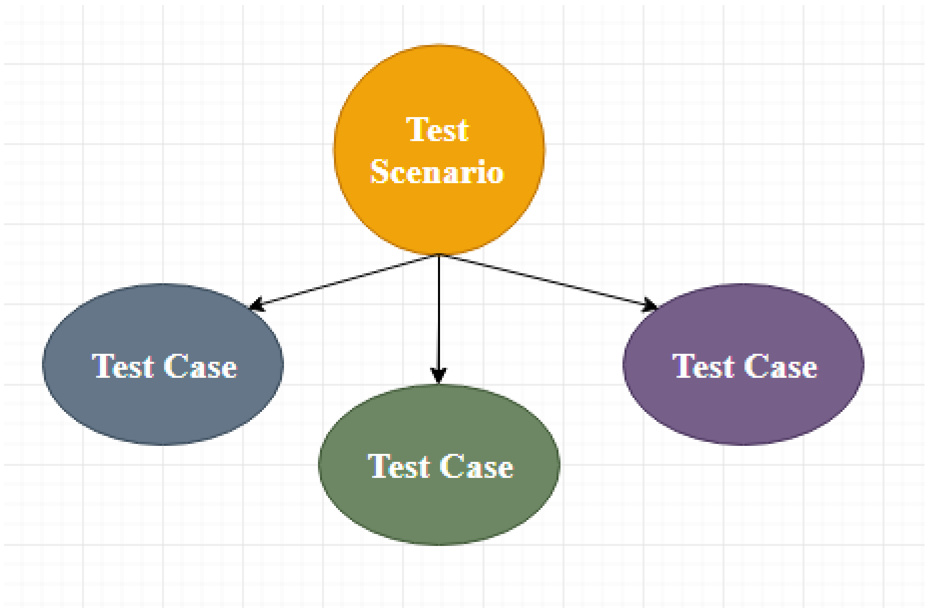
- ✓ Once you listed all the possible test scenarios (Test Conditions), create a traceability matrix to find out whether each and every requirement has a corresponding test scenario or not.
- ✓ Supervisor of the project reviews all scenarios. Later, they are evaluated by other stakeholders of the project.

- **Features of Test Scenario:-**
  - ✓ The test scenario is a liner statement that guides testers for the testing sequence.
  - ✓ Test scenario reduces the complexity and repetition of the product.
  - ✓ Test scenario means talking and thinking about tests in detail but write them in liner statements.
  - ✓ It is a thread of operations.
  - ✓ Test scenario becomes more important when the tester does not have enough time to write test cases, and team members agree with a detailed liner scenario.
  - ✓ The test scenario is a time saver activity.
  - ✓ It provides easy maintenance because the addition and modification of test scenarios are easy and independent.
- **Note:** Some rules have to be followed when we were writing test scenarios
  - ✓ Always list down the most commonly used feature (Functionality) and module by the users.
  - ✓ We always start the scenarios by picking module by module so that a proper sequence is followed as well as we don't miss out on any module level.
  - ✓ Generally, scenarios are module level for Functionality point of view.
  - ✓ Delete scenarios should always be the last option else, and we will waste lots of time in creating the data once again.
  - ✓ It should be written in a simple language.
  - ✓ Every scenario should be written in one line or a maximum of two lines and not in the paragraphs.
  - ✓ Every scenario should consist of Dos and checks/test.



TEST CASE

- The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.



- It is an in-details document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process.
- Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.
- Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.
- To write the test case, we must have the requirements to derive the inputs, and the test scenarios must be written so that we do not miss out on any features for testing. Then we should have the test case template to maintain the uniformity, or every test engineer follows the same approach to prepare the test document.
- Generally, we will write the test case whenever the developer is busy in writing the code.
- Once the test cases are created from the requirements, it is the job of the testers to execute those test cases. The testers read all the details in the test case, perform the test steps, and then based on the expected and actual result, mark the test case as **Pass** or **Fail**.

- **When do we write a test case?**

  We will write the test case when we get the following:

  - ✓ When the customer gives the business needs then, the developer starts developing and says that they need 3.5 months to build this product.
  - ✓ And In the meantime, the testing team will **start with writing the test cases**.
  - ✓ Once it is done, it will send it to the Test Lead for the review process.
  - ✓ And when the developers finish developing the product, it is handed over to the testing team.
- **Note:** When writing the test case, the actual result should never be written as the product is still being in the development process. That's why the actual result should be written only after the execution of the test cases.

- **TEST CASE TEMPLATE:-**
  - ✓ **Test case ID:-** A unique identifier of the test case. It is a mandatory field that uniquely identifies a test case e.g. TC_01. It is often generated automatically if creating test cases using tools. If it is assigned manually, it is advisable to make it meaningful to understand the purpose of a test case clearly.
  - ✓ **Test case Objective:-** Detailed description of the test case.
    This field defines the purpose of the test case e.g. To test/verify/check that the user can login with a valid username and valid password.
  - ✓ **Pre-condition:-** These are the necessary conditions that need to be satisfied by every testers before starting the test execution process. A set of prerequisites that must be followed before executing the test steps.
    For example – while testing the functionality of the application for login, then we can have the pre-requisite as "User must be registered in to the application".
  - ✓ **Step ID:-** An Identifier for each steps
  - ✓ **Step Description:-** Detailed steps for performing the test case.
    This is the most important field of a test case. The tester should aim to have clear and unambiguous steps in the test step description field so that some other person can follow the test steps during test execution.
  - ✓ **Input data:-** These are the values or the input we need to be created which will be used as an input for execution. For example – while testing the login functionality, the test data / Input data field can have the actual value of the username and password to be used during test execution.
  - ✓ **Expected Result:-** The expected result in order to pass the test.
    Based on the test steps followed and the test data used, we come up with the expected result e.g. the user should successfully login and navigated to home page.

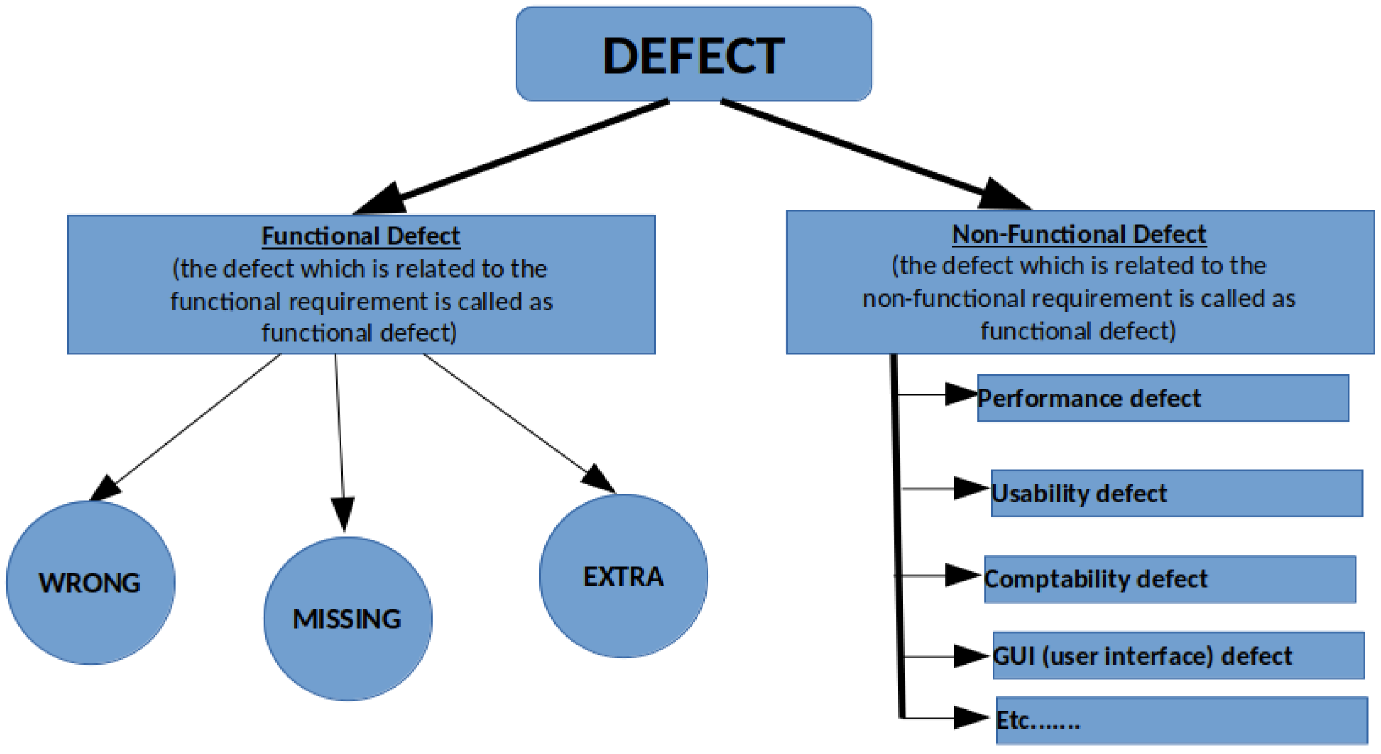
- ✓ **Actual Result:-**The actual result after executing the test steps.
  This field is filled during test execution only. In this field, we write the actual result observed during the test case execution.
- ✓ **Status:- Pass/Fail** status of the test execution.
  Based on the expected result and the actual result, the test case is marked as passed or Failed.
  Apart from Pass/Fail, we can have other values also like – **Deferred/Hold** (when the test case is marked to be executed later, for some reason) and **Blocked** (when the test case execution is blocked due to some other issue in the application).
- ✓ **Remark:-** This is an optional field where the tester may define the defect in details which they found during their execution for that particular test case. Here they may also inform about the date, time of the test case execution and the person the who executed the test case

-
-
-
- hno
-

CHAPTER 8
DEFECT MANAGEMENT

Defect:- the difference or variance between the excepted result and the actual result is called as the defect. There are 2 types of defect

a. **Functional defect**
b. **Non-Functional defect**


DEFECT
Functional Defect
(the defect which is related to the functional requirement is called as functional defect)
Non-Functional Defect
(the defect which is related to the non-functional requirement is called as functional defect)
WRONG
MISSING
EXTRA
Performance defect
Usability defect
Comptability defect
GUI (user interface) defect
Etc.......

Attributes of defect:-

1. Severity
2. Priority

Severity:-

1. Severity is defined as the degree of impact a Defect has on the Functionality (development or operation) of Software (a component application) being tested.
2. It is decided on the bases of impact or effect of defect on the functionality of the software.

Priority:-

1. Priority is defined as parameter that decides the order in which a defect should be fixed. Defect having the higher priority should be fixed first.
2. It is decided on the bases of impact or effect of defect on the business flow of the s/w.

Difference between Severity and Priority in Testing:

SEVERITY	PRIORITY
Severity is a parameter to denote the impact of a particular defect on the software.	Priority is a parameter to decide the order in which defects should be fixed.
Severity means how severe defect is affecting the functionality.	Priority means how fast defect has to be fixed.
Severity is related to the quality standard.	Priority is related to scheduling to resolve the problem.
Testing engineer (TESTER) decides the severity level of the defect.	Product manager / DEVELOPERS decide the priorities of defects. CLIENT may also decide the Priority
Its value is objective.	Its value is subjective.
Its value doesn't change from time to time.	Its value changes from time to time.
Severity is of 3 types: High, Medium and Low	Priority is of 3 types: : High, Medium and Low

SEVERITY	PRIORITY
High: - the s/w is in unusable state, abnormal termination, OS freeze , Major Functionalities not working, system crash, blocker defect, etc..	**High:-** the defect need to be fixed in immediately or on urgent bases
Medium: - the s/w is in usable state but some minor functionalities not working, Performance related issues, etc…	**Medium:-** the defect need to be fixed before release
Low: - GUI related issues, spelling mistake,(all non-functional related issues) etc…	**Low:-** the defect may be fixed if time permits or may not be fixed . it can be fixed in next release also

Combination of Severity and Priority :-

1. High Severity and High Priority:-

If the defect which effects or impacts on both functionality of the software as well as the business flow then it is called as HS &HP

2. High Severity and Low Priority:-

If the defect which effects or impacts on the functionality of the software but doesn't impact on the business flow then it is called as HS &LP

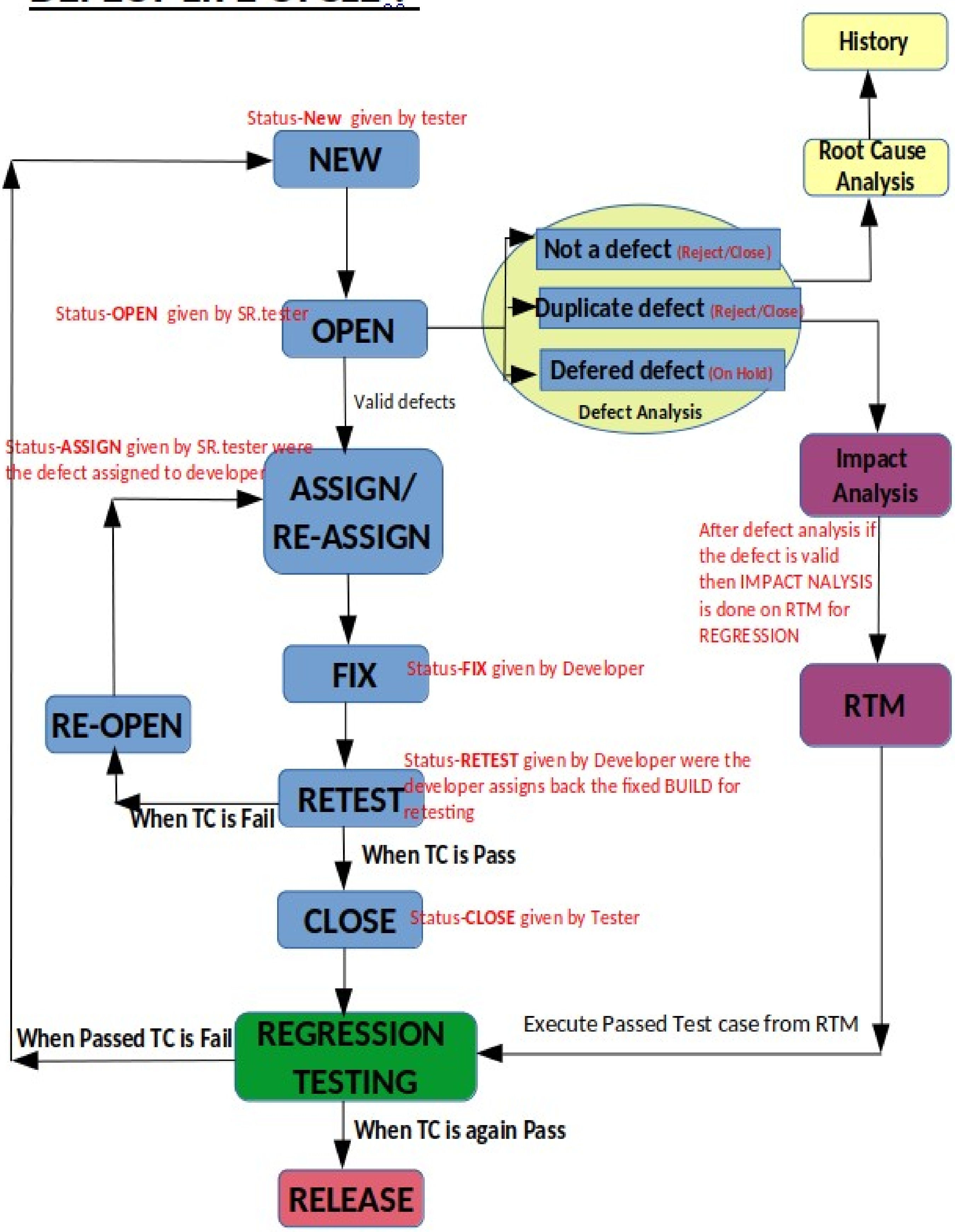
3. Low Severity and High Priority:-

If the defect which doesn't effects or impacts on the functionality of the software but impacts on the business flow then it is called as LS &HP

4. Low Severity and Low Priority:-

If the defect which doesn't effects or impacts on both functionality of the software as well as the business flow then it is called as LS &LP

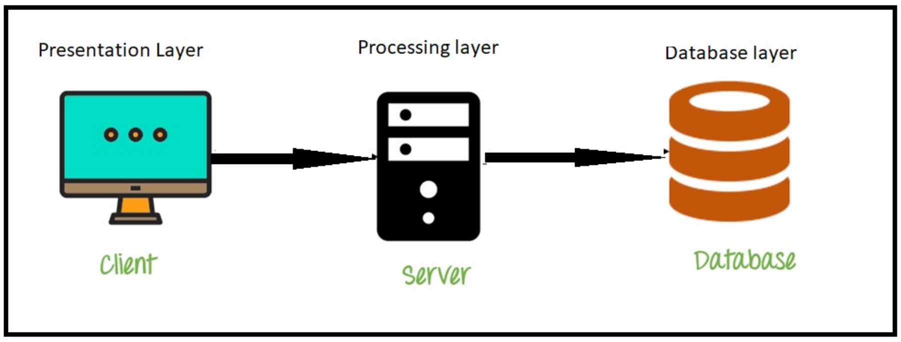
DEFECT LIFE CYCLE :-


History
Root Cause Analysis
Status-New given by tester
NEW
Not a defect (Reject/Close)
Duplicate defect (Reject/Close)
Defered defect (On Hold)
Defect Analysis
Status-OPEN given by SR.tester
OPEN
Valid defects
Impact Analysis
Status-ASSIGN given by SR.tester were the defect assigned to developer
ASSIGN/ RE-ASSIGN
After defect analysis if the defect is valid then IMPACT NALYSIS is done on RTM for REGRESSION
FIX
Status-FIX given by Developer
RE-OPEN
RTM
Status-RETEST given by Developer were the developer assigns back the fixed BUILD for retesting
RETEST
When TC is Fail
When TC is Pass
CLOSE
Status-CLOSE given by Tester
Execute Passed Test case from RTM
When Passed TC is Fail
REGRESSION TESTING
When TC is again Pass
RELEASE

CHAPTER 10
WEB APPLICATION TESTING

1 tier Architecture

The simplest of Database Architecture are **1 tier** where the Client, Server, and Database all reside on the same machine. Anytime you install a DB in your system and access it to practise SQL queries it is
1-tier architecture. But such architecture is rarely used in production.

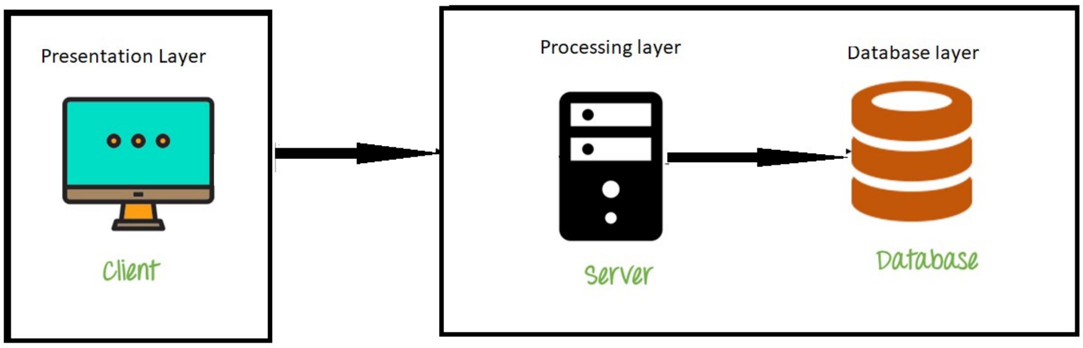


1-tier Architecture

2-tier Architecture

A two-tier architecture is a database architecture where

1. Presentation layer runs on a client (PC, Mobile, Tablet, etc)
2. Data is stored on a Server.



2-tier Architecure

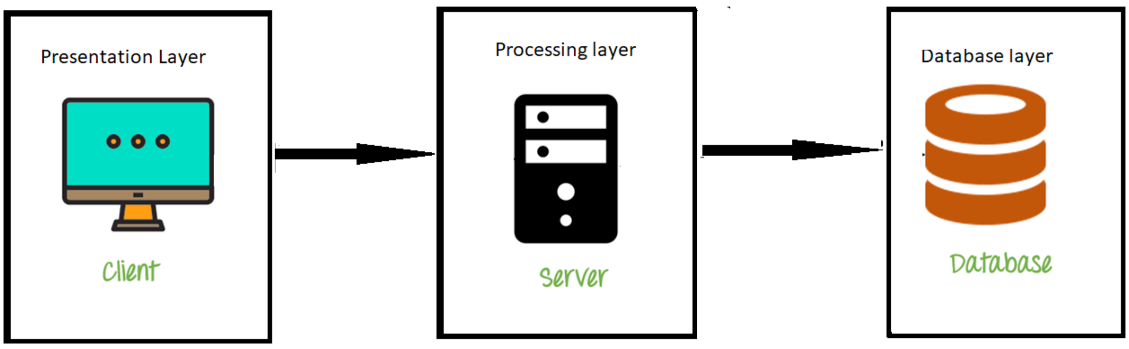
Here, the Presentation layer is present in one system and on the other hand processing layer AND Database layer is in another system then that is called as 2-tier Architecture

3-tier Architecture

3-tier schema is an extension of the 2-tier architecture. 3-tier architecture has following layers

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application(Processing) layer (server)
3. Database Server

Here, all the 3 layers i.e Presentation , Processing layer AND Database layer are different system in different location then that is called as 3-tier Architecture. It is also called as Web Application were it need internet to connect with all 3 components



3-teir Architecture

Testing which done on web application:-

1. **Security Testing:-**
    a. **Cookie Testing**
    b. **Authentication**
    c. **Authorization**
    d. **Password testing:-**
        i. The password should always be in the encrypted format.
        ii. The password field should not allow copy/paste option for password (copy/paste option should be disabled for password field)
    e. **url testing:-**
        i. When we try to open a url of any website in a browser which is copied from another browser then instead of opening the exact current page it should redirect to the Login page.
    f. **SQL Injection:-**
        i. SQL Injection is a hacking technique used by the hacker to get an Unauthorized access into the website by injecting a malicious query.
        ii. SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.
        iii. An SQL Injection vulnerability may affect any website or web application that uses an SQL database such as MySQL, Oracle, SQL Server, or others. Criminals may use it to gain unauthorized access to your sensitive data: customer information, personal data, trade secrets, intellectual property, and more. SQL Injection attacks are one of the

oldest, most prevalent, and most dangerous web application vulnerabilities.

2. **<u>Link Testing:-</u>**
    a. To check the broken link of a website (both internal link as well as external link)
    b. Types of Links
    Links are used to “link” a visitor from one area to another. There are many types of links :
        - Local: A page on the same server or directory
        - Internal: A section on the current page or document
        - External: A page or site on a different server or directory
        - Download: A file for the visitor to download
        - E-mail: Opens the visitor’s e-mail program

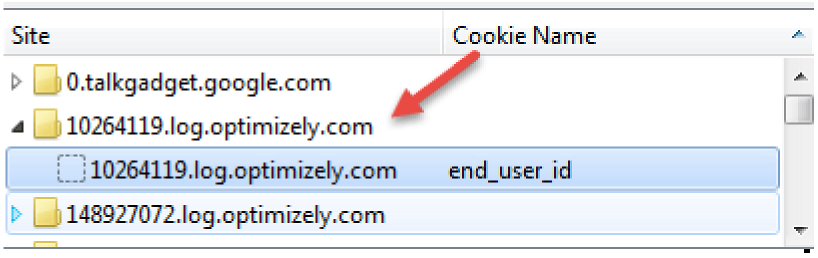
3. **Form Testing**
    a. Form level validation
    b. Field level validation
    c. Form navigation
    d. Data entry
    e. Error Checking
4. **<u>Reliability testing</u>**:- It is a software testing type, that checks whether the software can perform a failure-free operation for a specified period of time in a particular environment
5. **<u>Availability testing</u>**:- As a general idea, availability is a measure of how often the application is available for use. More specifically, availability is a percentage calculation based on how often the application is actually available to handle service requests when compared to the total, planned, available runtime. The formal calculation of availability includes repair time because an application that is being repaired is not available for use.
6. Performance testing
7. Compatibility testing
8. Usability testing
9. Localization testing

10. Internationalization testing
11. Retesting
12. Regression testing
13. Functional testing for correctness and completeness

What is Cookie Testing?

Cookie Testing is defined as a Software Testing type that checks Cookie stored in your web browser. A cookie is a small piece of information that is stored in a text file on user's (client) hard drive by the web server. Cookies are created by the Web Server and it is stored in the web Browser. This piece of information is then sent back to the server each time the browser requests a page from the server. Usually, cookie contains personalized user data or information that is used to communicate between different web pages.

The screen-shot below shows cookies for different websites.



Site	Cookie Name
0.talkgadget.google.com	
10264119.log.optimizely.com	
10264119.log.optimizely.com	end_user_id
148927072.log.optimizely.com	

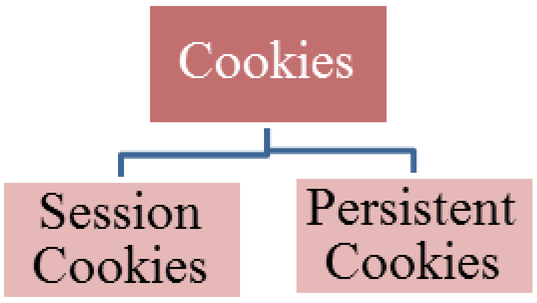
In other words, cookies are nothing but a user's identity and used to track where the user navigated throughout the pages of the website. The purpose of a cookie is to make rapid interaction between users and websites. Applications, where cookies can be used, is to implement a shopping cart, personalized web experience, user tracking, marketing, user sessions etc.

What is the Content of Cookie?

The cookie consists of mainly four things

1. The name of the server the cookie was sent from
2. Cookies Lifetime
3. A value. This is usually a randomly generated unique number
4. Information or data about the user details in an encrypted format

Types of Cookies



Usually, there are two types of cookies written on user machines

- **Session Cookies**: These cookies are active till the browser that triggers the cookie is open. When we close the browser this session cookie gets deleted automatically.
- **Persistent Cookies**: These cookies are written permanently on the user machine and it lasts for months or years. It has its own validity period/Expiration date. Once the validity is expired it is deleted automatically.

Where Cookies are stored?

When any web page application writes a cookie, it is stored in a text file on user hard disk drive. The path where the cookies are saved depends on the browser. **Different browsers store cookie in different paths.**

For example, in **Mozilla Firefox** browser you can see the cookies in browser options. To view this click on Tools **->** Options **->** Privacy and then click on "Remove Individual Cookies".

How to test Cookies (Cookie Testing)– Sample Test Cases

Following is an important checklist to test cookies in Software Engineering

1. **Disabling cookies:** Disable all cookies and attempt to use the site's major functions.
2. **Cookies information is in encryption or not:** To check whether Sensitive information like passwords and usernames should be encrypted or not
3. **Cookie testing with multiple browsers:** Check your website page is writing the cookies properly on a different browser as expected

4. **Selectively rejecting cookies:** Delete all the cookies for the websites and see how the website reacts to it
5. **Access to cookies:** Cookies written by one website should not be accessible by others
6. **Testing the validation as per the types of cookie been stored:** Testing should be done properly to check the validity of that cookie for both persistent or session cookie