

CCS366 SOFTWARE TESTING AND AUTOMATION

UNIT I FOUNDATIONS OF SOFTWARE TESTING 6

Why do we test Software?, Black-Box Testing and White-Box Testing, Software Testing Life Cycle, V-model of Software Testing, Program Correctness and Verification, Reliability versus Safety, Failures, Errors and Faults (Defects), Software Testing Principles, Program Inspections, Stages of Testing: Unit Testing, Integration Testing, System Testing

UNIT II TEST PLANNING 6

The Goal of Test Planning, High Level Expectations, Intergroup Responsibilities, Test Phases, Test Strategy, Resource Requirements, Tester Assignments, Test Schedule, Test Cases, Bug Reporting, Metrics and Statistics.

UNIT III TEST DESIGN AND EXECUTION 6

Test Objective Identification, Test Design Factors, Requirement identification, Testable Requirements, Modeling a Test Design Process, Modeling Test Results, Boundary Value Testing, Equivalence Class Testing, Path Testing, Data Flow Testing, Test Design Preparedness Metrics, Test Case Design Effectiveness, Model-Driven Test Design, Test Procedures, Test Case Organization and Tracking, Bug Reporting, Bug Life Cycle.

UNIT IV ADVANCED TESTING CONCEPTS 6

Performance Testing: Load Testing, Stress Testing, Volume Testing, Fail-Over Testing, Recovery Testing, Configuration Testing, Compatibility Testing, Usability Testing, Testing the Documentation, Security testing, Testing in the Agile Environment, Testing Web and Mobile Applications

UNIT V TEST AUTOMATION AND TOOLS 6

Automated Software Testing, Automate Testing of Web Applications, Selenium: Introducing Web Driver and Web Elements, Locating Web Elements, Actions on Web Elements, Different Web Drivers, Understanding Web Driver Events, Testing: Understanding Testing.xml, Adding Classes, Packages, Methods to Test, Test Reports

30 PERIODS**PRACTICAL EXERCISES: 30 PERIODS**

1. Develop the test plan for testing an e-commerce web/mobile application (www.amazon.in).
2. Design the test cases for testing the e-commerce application
3. Test the e-commerce application and report the defects in it.
4. Develop the test plan and design the test cases for an inventory control system.
5. Execute the test cases against a client server or desktop application and identify the defects.
6. Test the performance of the e-commerce application.
7. Automate the testing of e-commerce applications using Selenium.
8. Integrate TestNG with the above test automation.
9. Mini Project:
 - a) Build a data-driven framework using Selenium and TestNG
 - b) Build Page object Model using Selenium and TestNG
 - c) Build BDD framework with Selenium, TestNG and Cucumber

TOTAL:60 PERIODS**TEXTBOOKS**

1. Yogesh Singh, "Software Testing", Cambridge University Press, 2012
2. Unmesh Gundecha, Satya Avasarala, "Selenium WebDriver 3 Practical Guide" - Second Edition 2018

REFERENCES

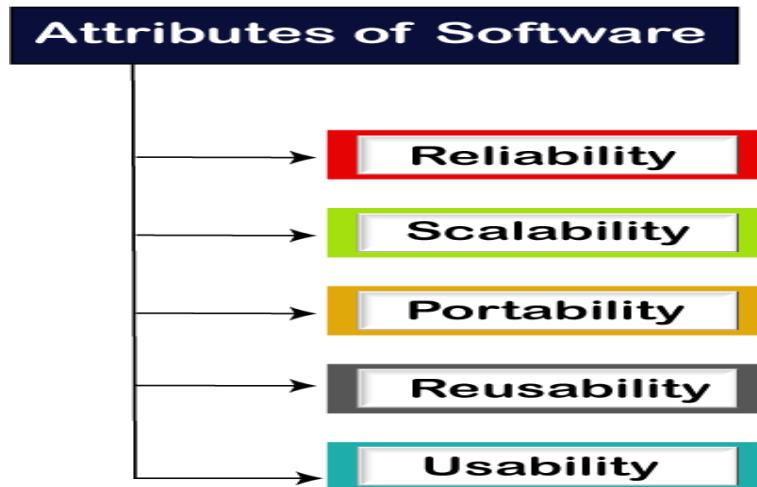
1. Glenford J. Myers, Corey Sandler, Tom Badgett, The Art of Software Testing, 3rd Edition, 2012, John Wiley & Sons, Inc.
2. Ron Patton, Software testing, 2nd Edition, 2006, Sams Publishing
3. Paul C. Jorgensen, Software Testing: A Craftsman's Approach, Fourth Edition, 2014, Taylor & Francis Group.
4. Carl Cocchiaro, Selenium Framework Design in Data-Driven Testing, 2018, Packt Publishing.
5. Elfriede Dustin, Thom Garrett, Bernie Gaurf, Implementing Automated Software Testing, 2009, Pearson Education, Inc.
6. Satya Avasarala, Selenium WebDriver Practical Guide, 2014, Packt Publishing.
7. Varun Menon, TestNg Beginner's Guide, 2013, Packt Publishing.

UNIT I FOUNDATIONS OF SOFTWARE TESTING**6**

Why do we test Software?, Black-Box Testing and White-Box Testing, Software Testing Life Cycle, V-model of Software Testing, Program Correctness and Verification, Reliability versus Safety, Failures, Errors and Faults (Defects), Software Testing Principles, Program Inspections, Stages of Testing: Unit Testing, Integration Testing, System Testing

Why do we test Software?

. Software testing is a process of identifying the correctness of software by considering its all attributes (Reliability, Scalability, Portability, Re-usability, Usability) and evaluating the execution of software components to find the software bugs or errors or defects



Software testing provides an independent view and objective of the software and gives surety of fitness of the software. It involves testing of all components under the required services to confirm that whether it is satisfying the specified requirements or not. The process is also providing the client with information about the quality of the software.

Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

What is Testing

Testing is a group of techniques to determine the correctness of the application under the predefined script but, testing cannot find all the defect of application. The main intent of testing is to detect failures of the application so that failures can be discovered and corrected. It does not demonstrate that a product functions properly under all conditions but only that it is not working in some specific conditions.

Testing furnishes comparison that compares the behavior and state of software against mechanisms because the problem can be recognized by the mechanism. The mechanism may include past versions of the same specified product, comparable products, and interfaces of expected purpose, relevant standards, or other criteria but not limited up to these.

Testing includes an examination of code and also the execution of code in various environments, conditions as well as all the examining aspects of the code. In the current scenario of software development, a testing team may be separate from the development team so that information derived from testing can be used to correct the process of software development.

The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc. For example, the audience of banking is totally different from the audience of a video game. Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

black-box testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Test procedure

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality.

It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique. All these techniques have been explained in detail within the tutorial.

Test cases

Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications. For the testing, the test designer selects both positive test scenario by taking valid input values and adverse test scenario by taking invalid input values to determine the correct output. Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team, there is not any involvement of the development team of software.

Techniques Used in Black Box Testing

<u>Decision Table Technique</u>	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
<u>Boundary Value Technique</u>	Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
<u>State Transition Technique</u>	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
<u>All-pair Testing Technique</u>	All-pair testing Technique is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
<u>Cause-Effect Technique</u>	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result. It is based on a collection of requirements.
<u>Equivalence Partitioning Technique</u>	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.

<u>Error Guessing Technique</u>	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
<u>Use Case Technique</u>	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end

White-Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

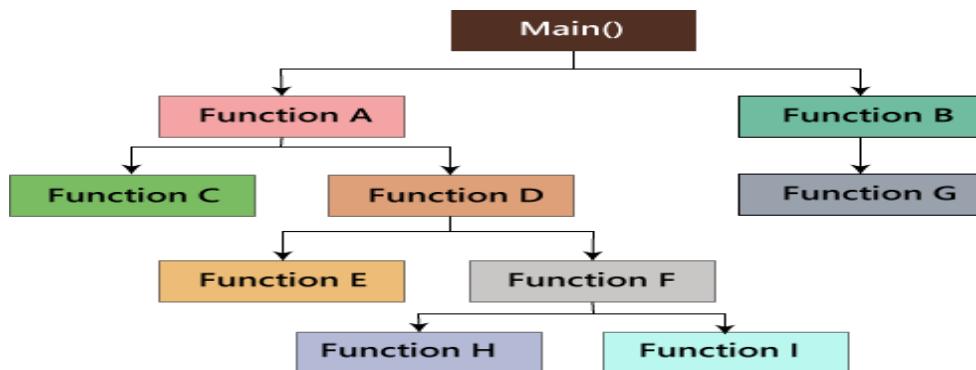
- Fixing the bug might interrupt the other features. Therefore, the test engineers should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program

Path testing

In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



And test all the independent paths implies that suppose a path from main() to function G, first set the parameters and test if the program is correct in that particular path, and in the same way test all other paths and fix the bugs.

Loop testing

In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.

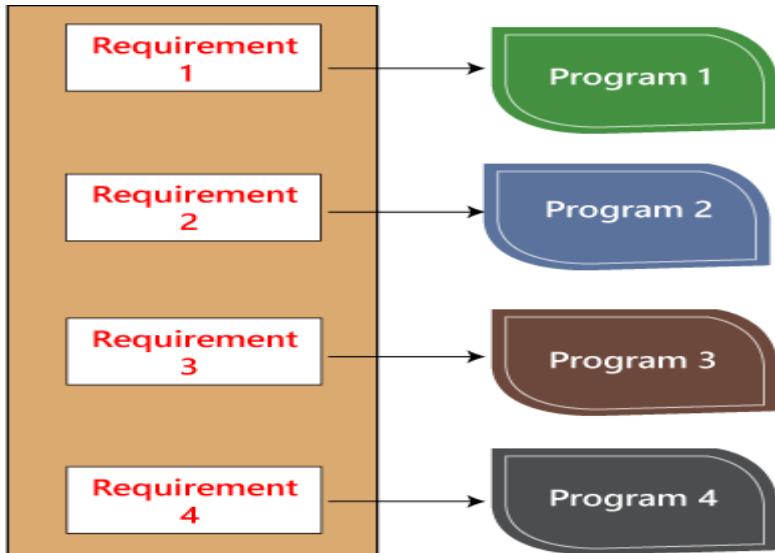
For example: we have one program where the developers have given about 50,000 loops.

```
1.      {
2.      while(50,000)
3.      ....
4.      ....
5.      }
```

We cannot test this program manually for all the 50,000 loops cycle. So we write a small program that helps for all 50,000 cycles, as we can see in the below program, that test P is written in the similar language as the source code program, and this is known as a Unit test. And it is written by the developers only.

```
1.      Test P
2.      {
3.      ....
4.      .... }
```

As we can see in the below image that, we have various requirements such as 1, 2, 3, 4. And then, the developer writes the programs such as program 1,2,3,4 for the parallel conditions. Here the application contains the 100s line of codes.

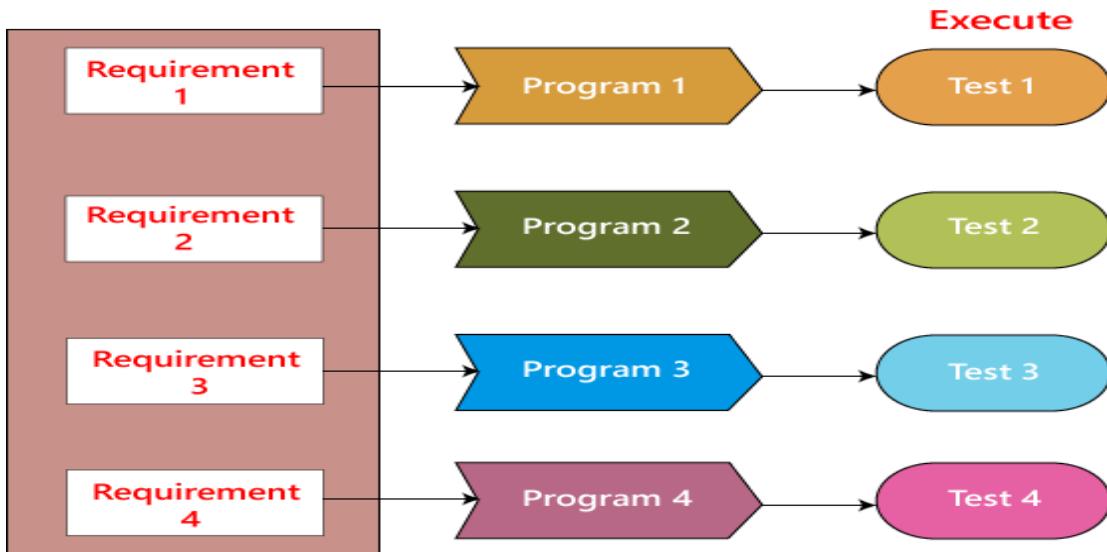


The developer will do the white box testing, and they will test all the five programs line by line of code to find the bug. If they found any bug in any of the programs, they will correct it. And they again have to test the system then this process contains lots of time and effort and slows down the product release time.

Now, suppose we have another case, where the clients want to modify the requirements, then the developer will do the required changes and test all four program again, which take lots of time and efforts.

These issues can be resolved in the following ways:

In this, we will write test for a similar program where the developer writes these test code in the related language as the source code. Then they execute these test code, which is also known as **unit test programs**. These test programs linked to the main program and implemented as programs.



Therefore, if there is any requirement of modification or bug in the code, then the developer makes the adjustment both in the main program and the test program and then executes the test program.

Condition testing

In this, we will test all logical conditions for both **true** and **false** values; that is, we will verify for both **if** and **else** condition.

For example:

- ```
1. if(condition) - true
2. {
3.
4.
5.
6. }
7. else - false
8. {
9.
```

```
10.
11.
12. }
```

The above program will work fine for both the conditions, which means that if the condition is accurate, and then else should be false and conversely.

### Testing based on the memory (size) perspective

The size of the code is increasing for the following reasons:

- **The reuse of code is not there:** let us take one example, where we have four programs of the same application, and the first ten lines of the program are similar. We can write these ten lines as a discrete function, and it should be accessible by the above four programs as well. And also, if any bug is there, we can modify the line of code in the function rather than the entire code.
- **The developers use the logic** that might be modified. If one programmer writes code and the file size is up to 250kb, then another programmer could write a similar code using the different logic, and the file size is up to 100kb.
- **The developer declares so many functions and variables** that might never be used in any portion of the code. Therefore, the size of the program will increase.

For example,

```
1. Int a=15;
2. Int b=20;
3. String S= "Welcome";
4.
5.
6.
7.
8.
9. Int p=b;
10. Create user()
11. {
12.
```

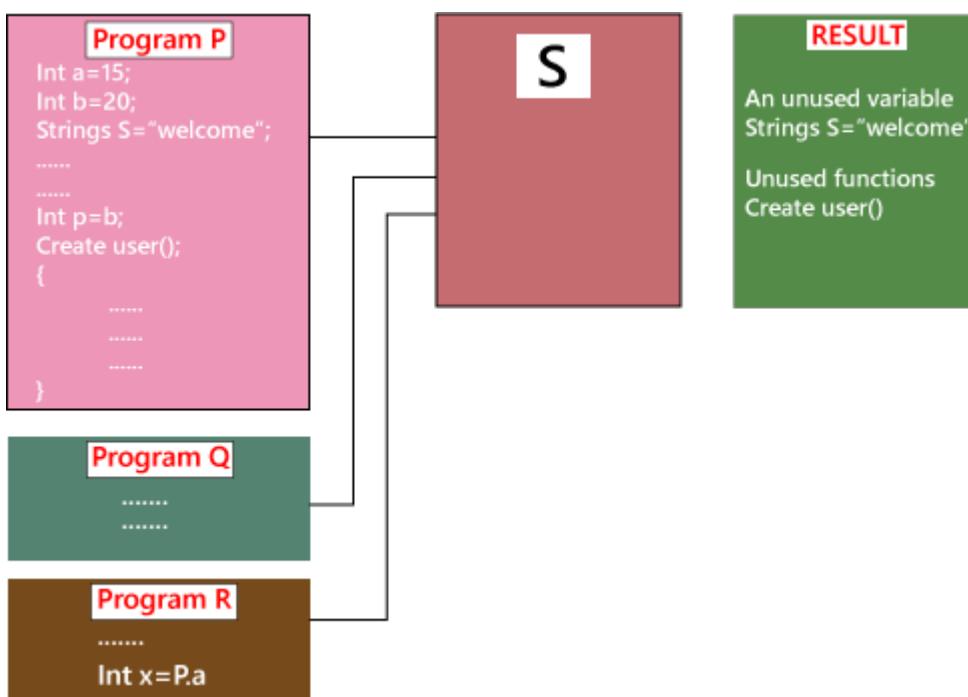
13. .....

14. 200's line of code

15. }

In the above code, we can see that the **integer a** has never been called anywhere in the program, and also the function **Create user** has never been called anywhere in the code. Therefore, it leads us to memory consumption.

We cannot remember this type of mistake manually by verifying the code because of the large code. So, we have a built-in tool, which helps us to test the needless variables and functions. And, here we have the tool called **Rational purify**.



Suppose we have three programs such as Program P, Q, and R, which provides the input to S. And S goes into the programs and verifies the unused variables and then gives the outcome. After that, the developers will click on several results and call or remove the unnecessary function and the variables.

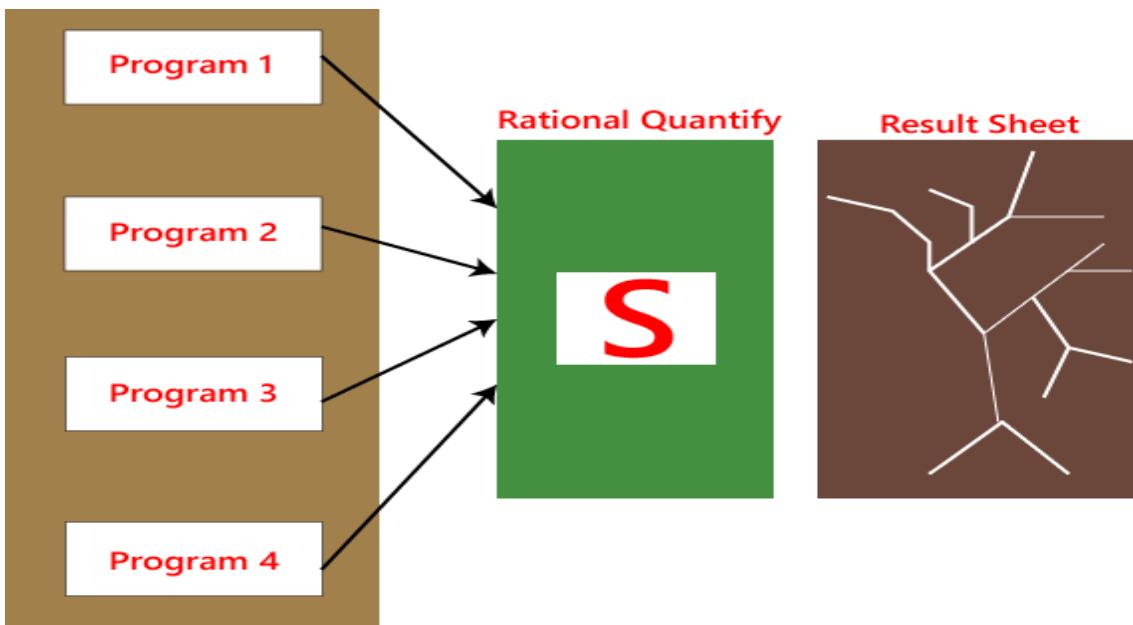
This tool is only used for the [C programming language](#) and [C++ programming language](#); for another language, we have other related tools available in the market.

- o The developer does not use the available in-built functions; instead they write the full features using their logic. Therefore, it leads us to waste of time and also postpone the product releases.

**Test the performance (Speed, response time) of the program**

The application could be slow for the following reasons:

- When logic is used.
- For the conditional cases, we will use **or & and** adequately.
- Switch case, which means we cannot use **nested if**, instead of using a switch case.

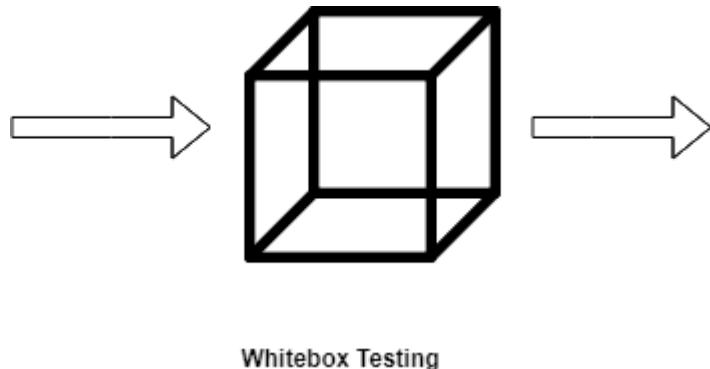


As we know that the developer is performing white box testing, they understand that the code is running slow, or the performance of the program is also getting deliberate. And the developer cannot go manually over the program and verify which line of the code is slowing the program.

To recover with this condition, we have a tool called **Rational Quantify**, which resolves these kinds of issues automatically. Once the entire code is ready, the rational quantify tool will go through the code and execute it. And we can see the outcome in the result sheet in the form of thick and thin lines.

Here, the thick line specifies which section of code is time-consuming. When we double-click on the thick line, the tool will take us to that line or piece of code automatically, which is also displayed in a different color. We can change that code and again and use this tool. When the order of lines is all thin, we know that the presentation of the program has enhanced. And the developers will perform the white box testing automatically because it saves time rather than performing manually.

Test cases for white box testing are derived from the design phase of the software development lifecycle. Data flow testing, control flow testing, path testing, branch testing, statement and decision coverage all these techniques used by white box testing as a guideline to create an error-free software.



White box testing follows some working steps to make testing manageable and easy to understand what the next task to do. There are some basic steps to perform white box testing.

#### Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
- In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.
- This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

#### Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.

- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

### Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

### Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

### Techniques Used in White Box Testing

|                             |                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>Data Flow Testing</u>    | Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.                                                                                                                                                                                                        |
| <u>Control Flow Testing</u> | Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program. |
| <u>Branch Testing</u>       | Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.                                                                                                                                                                                              |

|                          |                                                                                                                                                                                                                                                                                                                                |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>Statement Testing</u> | Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.                             |
| <u>Decision Testing</u>  | This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false. |

### Difference between white-box testing and black-box testing

Following are the significant differences between white box testing and black box testing:

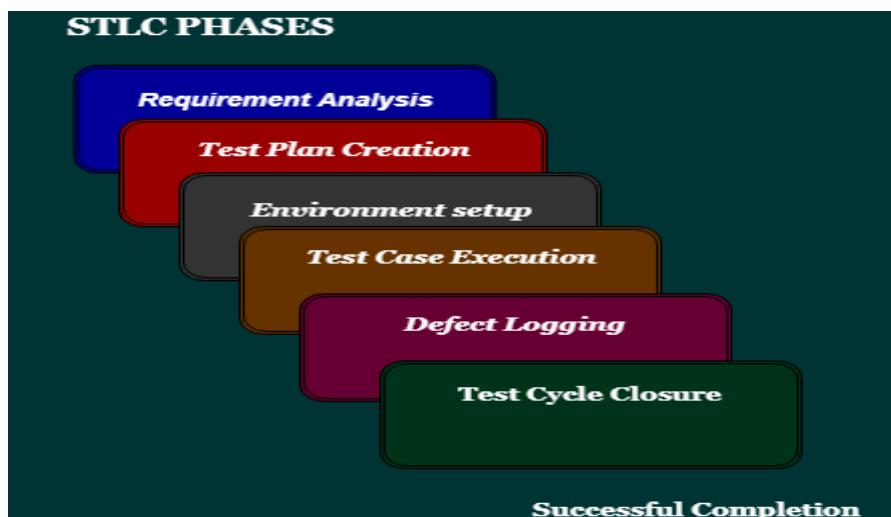
| White-box testing                                                             | Black box testing                                                                                    |
|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| The developers can perform white box testing.                                 | The test engineers perform the black box testing.                                                    |
| To perform WBT, we should have an understanding of the programming languages. | To perform BBT, there is no need to have an understanding of the programming languages.              |
| In this, we will look into the source code and test the logic of the code.    | In this, we will verify the functionality of the application based on the requirement specification. |
| In this, the developer should know about the internal design of the code.     | In this, there is no need to know about the internal design of the code.                             |

### Software Testing Life Cycle

The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process. The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product.

### **Software testing life cycle contains the following steps:**

1. Requirement Analysis
2. Test Plan Creation
3. Environment setup
4. Test case Execution
5. Defect Logging
6. Test Cycle Closure



### **Requirement Analysis:**

The first step of the manual testing procedure is requirement analysis. In this phase, tester analyses requirement document of SDLC (Software Development Life Cycle) to examine requirements stated by the client. After examining the requirements, the tester makes a test plan to check whether the software is meeting the requirements or not.

| Entry Criteria                                                                             | Activities                                                                                             | Deliverable                                                            |
|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
| For the planning of test plan requirement specification, application architecture document | Prepare the list of all requirements and queries, and get resolved from Technical Manager/Lead, System | List of all the necessary tests for the testable requirements and Test |

|                                                           |                                                                                                                                                                                                                                                                        |                     |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| and well-defined acceptance criteria should be available. | <p>Architecture, Business Analyst and Client.</p> <p>Make a list of all types of tests (Performance, Functional and security) to be performed.</p> <p>Make a list of test environment details, which should contain all the necessary tools to execute test cases.</p> | environment details |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|

### Test Plan Creation:

Test plan creation is the crucial phase of STLC where all the testing strategies are defined. Tester determines the estimated effort and cost of the entire project. This phase takes place after the successful completion of the **Requirement Analysis Phase**. Testing strategy and effort estimation documents provided by this phase. Test case execution can be started after the successful completion of Test Plan Creation.

| Entry Criteria       | Activities                                                                                                                                                                                                                                                                                                                                                                    | Deliverable                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| Requirement Document | <p>Define Objective as well as the scope of the software.</p> <p>List down methods involved in testing.</p> <p>Overview of the testing process.</p> <p>Settlement of testing environment.</p> <p>Preparation of the test schedules and control procedures.</p> <p>Determination of roles and responsibilities.</p> <p>List down testing deliverables, define risk if any.</p> | <p>Test strategy document.</p> <p>Testing Effort estimation documents are the deliverables of this phase.</p> |

### Environment setup:

Setup of the test environment is an independent activity and can be started along with **Test Case Development**. This is an essential part of the manual testing procedure as without environment testing is not possible. Environment setup requires a group of essential software and hardware to create a test environment. The testing team is not involved in setting up the testing environment, its senior developers who create it.

| Entry Criteria                                                                | Activities                                                                                                                                                                                                 | Deliverable                         |
|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| Test strategy and test plan document.<br>Test case document.<br>Testing data. | Prepare the list of software and hardware by analyzing requirement specification.<br>After the setup of the test environment, execute the smoke test cases to check the readiness of the test environment. | Execution report.<br>Defect report. |

### Test case Execution:

Test case Execution takes place after the successful completion of test planning. In this phase, the testing team starts case development and execution activity. The testing team writes down the detailed test cases, also prepares the test data if required. The prepared test cases are reviewed by peer members of the team or Quality Assurance leader.

RTM (Requirement Traceability Matrix) is also prepared in this phase. Requirement Traceability Matrix is industry level format, used for tracking requirements. Each test case is mapped with the requirement specification. Backward & forward traceability can be done via RTM.

| Entry Criteria       | Activities                                                                                              | Deliverable                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Requirement Document | Creation of test cases.<br>Execution of test cases.<br>Mapping of test cases according to requirements. | Test execution result.<br>List of functions with the detailed explanation of defects. |

### Defect Logging:

Testers and developers evaluate the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. This phase determines the characteristics and drawbacks of the software. Test cases and bug reports are analyzed in depth to detect the type of defect and its severity.

Defect logging analysis mainly works to find out defect distribution depending upon severity and types. If any defect is detected, then the software is returned to the development team to fix the defect, then the software is re-tested on all aspects of the testing.

Once the test cycle is fully completed then test closure report, and test metrics are prepared.

| <b>Entry Criteria</b>                         | <b>Activities</b>                                                                                                                                                                                                                                   | <b>Deliverable</b>             |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| All document and reports related to software. | Evaluates the strategy of development, testing procedure, possible defects to use these practices in the future if there is a software with the same specification                                                                                  | Test closure report            |
| <b>Entry Criteria</b>                         | <b>Activities</b>                                                                                                                                                                                                                                   | <b>Deliverable</b>             |
| Test case execution report.<br>Defect report  | It evaluates the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives.<br><br>Defect logging analysis finds out defect distribution by categorizing in types and severity. | Closure report<br>Test metrics |

### **Test Cycle Closure:**

The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.

This phase evaluates the strategy of development, testing procedure, possible defects in order to use these practices in the future if there is a software with the same specification.

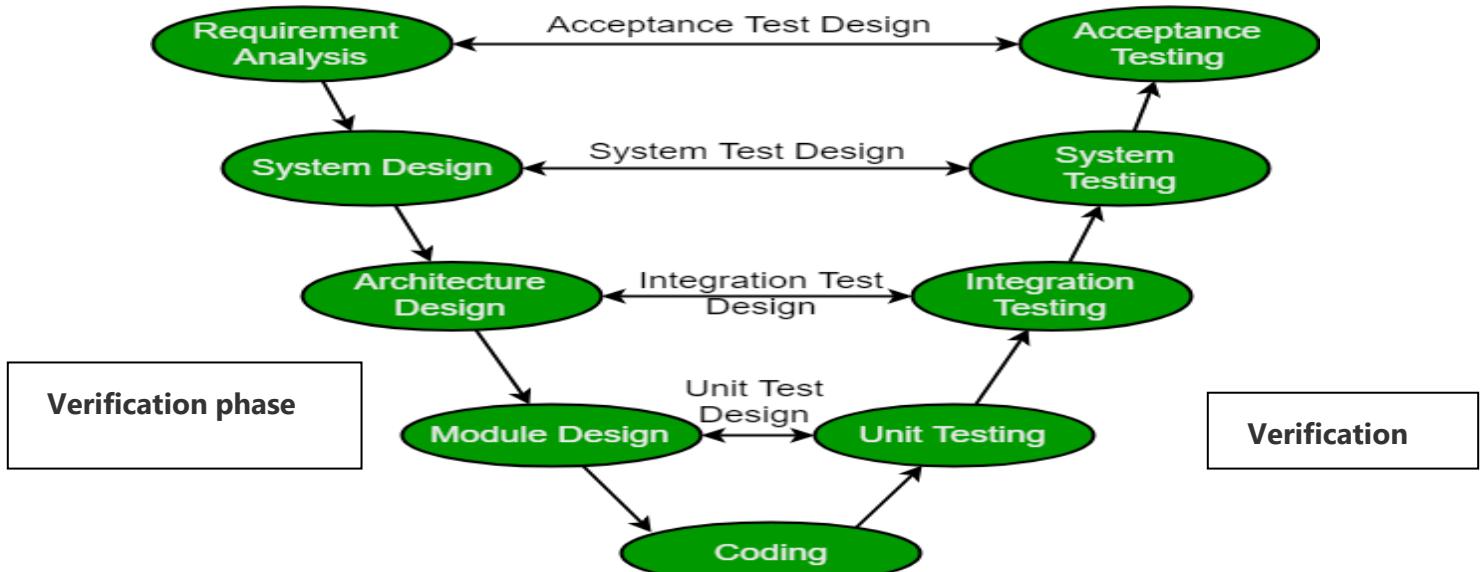
## V-model of Software Testing

**There are the various phases of Verification Phase of V-model:**

1. **Business requirement analysis:** This is the first step where product requirements understood from the customer's side. This phase contains detailed

Developers' life cycle

Tester's life cycle



communication to understand customer's expectations and exact requirements.

2. **System Design:** In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.

3. **Architecture Design:** The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc. The integration testing model is carried out in a particular phase.
4. **Module Design:** In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design
5. **Coding Phase:** After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

**There are the various phases of Validation Phase of V-model:**

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.
2. **Integration Testing:** Integration Test Plans are developed during the Architectural Design Phase. These tests verify that groups created and tested independently can coexist and communicate among themselves.
3. **System Testing:** System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Tests Plans are composed by the client's business team. System Test ensures that expectations from an application developer are met.
4. **Acceptance Testing:** Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

### When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

### Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

### Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

### Program Correctness and Verification

Definition. Software correctness testing is. [t]he process of executing a program with the intent of finding errors and is aimed primarily at improving quality assurance, verifying and validating described functionality, or estimating reliability.

### Important rules:

Below are some of the important rules for effective programming which are consequences of the program correctness theory.

- Defining the problem completely.
- Develop the algorithm and then the program logic.
- Reuse the proved models as much as possible.
- Prove the correctness of algorithms during the design phase.
- Developers should pay attention to the clarity and simplicity of your program.
- Verifying each part of a program as soon as it is developed.

Verification evaluates software artifacts (such as requirements, design, code, etc.) to ensure they meet the specified requirements and standards. It ensures the software is built according to the needs and design specifications. Validation evaluates software to meet the user's needs and requirements

Program correctness and verification are important aspects of software development and aim to ensure that a program behaves as intended and produces correct results. Verification involves the process of evaluating a program or system to determine if it meets specified requirements, while correctness refers to the program's adherence to its intended behavior.

There are several approaches to program correctness and verification, including:

**Testing:** Testing involves executing a program with various inputs and checking if the outputs match the expected results. Test cases are designed to cover different scenarios and edge cases to uncover potential bugs or errors. While testing can help identify defects, it does not guarantee the absence of errors in the program.

**Static Analysis:** Static analysis involves examining the program's source code or intermediate representation without executing it. Various techniques are used to analyze the code for potential issues, such as syntax errors, type violations, or common programming mistakes. Static analysis tools can help identify potential problems early in the development process.

**Formal Methods:** Formal methods involve the use of mathematical techniques to formally specify, model, and reason about the behavior of a program. This includes

techniques such as formal verification, where mathematical proofs are used to demonstrate the correctness of a program with respect to its specification. Formal methods are typically used for critical systems where high assurance and reliability are required.

**Model Checking:** Model checking is a technique that involves exhaustively exploring the state space of a system to verify if certain properties hold. It is particularly useful for systems with finite state spaces. Model checking tools can automatically explore all possible system states and verify properties specified in temporal logic, such as safety and liveness properties.

**Design by Contract:** Design by Contract is an approach where a program's behavior is specified using preconditions, postconditions, and invariants. Preconditions define the conditions that must hold before a function is executed, postconditions specify the expected behavior and output after execution, and invariants are properties that hold throughout the execution of a program. Design by Contract allows for runtime checks to verify if the program adheres to its contract.

**Code Reviews and Inspections:** Code reviews involve manual examination of the source code by developers or peers to identify potential defects, such as coding errors, logic flaws, or violations of coding standards. Code inspections can be performed using checklists or guidelines to ensure consistent quality and adherence to best practices.

Program correctness and verification techniques can be used individually or in combination to increase the confidence in the reliability and correctness of software systems. The choice of approach depends on the nature of the system, its criticality, and the resources available for verification.

## **Reliability versus Safety**

Reliability is the probability that a system or component will perform its intended function for a prescribed time and under stipulated environmental conditions. Reliability may thus be determined by the probability of failure per demand, whilst safety is also determined by the consequences of these failures.

Reliability and safety are both important aspects of software and system development, but they focus on different aspects of the system.

Reliability refers to the ability of a system to perform its intended function consistently and predictably over time. It involves ensuring that the system operates without failures, errors, or unexpected behavior. Reliability is often measured in terms of metrics such as availability, mean time between failures (MTBF), mean time to repair (MTTR), and overall system uptime.

Safety, on the other hand, concerns the protection of users, operators, and the environment from harm or damage caused by the system's operation. Safety is especially crucial in critical systems, such as those used in aviation, healthcare, nuclear power plants, and transportation. It involves identifying potential hazards, assessing risks, and implementing measures to prevent accidents or mitigate their consequences.

While reliability and safety are related, they have different priorities and requirements:

Reliability may focus on ensuring that a system operates correctly under normal conditions, delivering the expected results consistently. It involves techniques like testing, monitoring, fault tolerance, and error handling to minimize failures and maximize system availability.

Safety, on the other hand, emphasizes the identification and prevention of hazards, as well as the mitigation of risks associated with the system's operation. Safety-critical systems often employ techniques such as fault tolerance, redundancy, safety analysis, and compliance with safety standards and regulations.

In some cases, achieving high reliability can contribute to safety. For example, redundant systems or error-detection mechanisms can improve both reliability and safety by providing backup options or alerting users to potential hazards. However, a reliable system may still have safety concerns if it does not adequately address potential hazards or risks.

It's important to note that safety-critical systems require a more rigorous approach to verification, validation, and certification compared to non-critical systems. They often involve specialized techniques, such as formal methods, extensive testing, simulation, risk analysis, and compliance with safety standards and regulations.

In summary, reliability focuses on consistent and predictable system operation, while safety concentrates on preventing harm and minimizing risks. Both aspects are crucial for building trustworthy and robust systems, but safety-critical systems require additional measures and considerations to ensure the well-being of users and the environment.

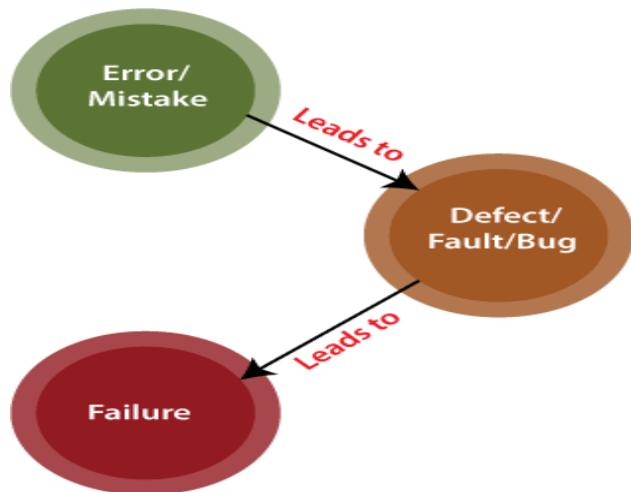
### Failures, Errors and Faults (Defects)

Many defects lead to the **software's failure**, which means that a loss specifies a fatal issue in software/ application or in its module, which makes the system unresponsive or broken. In other words, we can say that if an end-user detects an issue in the product, then that particular issue is called a **failure**.

Possibilities are there one defect that might lead to one failure or several failures. **For example**, in a bank application if the Amount Transfer module is not working for end-users when the end-user tries to transfer **money**, submit button is not working. Hence, this is a **failure**.

The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The **developers** use the term **error**.

### Failures, Errors and Faults (Defects)



The fault may occur in software because it has not added the code for fault tolerance, making an application act up.

fault may happen in a program because of the following reasons

- o A lack of resources
- o An invalid step
- o Inappropriate data definition

### Bug Vs. Defect Vs. Error Vs. Fault Vs. Failure

| Comparison basis  | Bug                                             | Defect                                                                                | Error                                                                                        | Fault                                                                                              | Failure                                                                     |
|-------------------|-------------------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| <b>Definition</b> | It is an informal name specified to the defect. | The <b>Defect</b> is the difference between the actual outcomes and expected outputs. | An <b>Error</b> is a mistake made in the code; that's why we cannot execute or compile code. | The <b>Fault</b> is a state that causes the software to fail to accomplish its essential function. | If the software has lots of defects, it leads to failure or causes failure. |

|                        |                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                              |                                                                                                                                                        |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Raised by</b>       | The <b>Test Engineers</b> submit the bug.                                                                                                                  | The <b>Testers</b> identify the defect. And it was also solved by the developer in the development phase or stage.                                                                                                                                                                                                | The <b>Developers and automation testengineers</b> raise the error.                                                                                                                                                                                                                    | <b>Human mistakes</b> cause fault.                                                                                                                                                                                                                                           | The failure finds by the manual test engineer through the <b>development cycle</b> .                                                                   |
| <b>Different types</b> | Different type of bugs are as follows: <ul style="list-style-type: none"> <li>○ Logic bugs</li> <li>○ Algorithmic bugs</li> <li>○ Resource bugs</li> </ul> | Different type of Defects are as follows:<br>Based on <b>priority</b> : <ul style="list-style-type: none"> <li>○ High</li> <li>○ Medium</li> <li>○ Low</li> </ul><br>And based on the severity: <ul style="list-style-type: none"> <li>○ Critical</li> <li>○ Major</li> <li>○ Minor</li> <li>○ Trivial</li> </ul> | Different type of Error is as below: <ul style="list-style-type: none"> <li>○ Syntactic Error</li> <li>○ User interface error</li> <li>○ Flow control error</li> <li>○ Error handling error</li> <li>○ Calculation error</li> <li>○ Hardware error</li> <li>○ Testing error</li> </ul> | Different type of Fault are as follows: <ul style="list-style-type: none"> <li>○ Business Logic Faults</li> <li>○ Functional and Logical Faults</li> <li>○ Faulty GUI</li> <li>○ Performance Faults</li> <li>○ Security Faults</li> <li>○ Software/hardware fault</li> </ul> | -----                                                                                                                                                  |
| <b>Reasons behind</b>  | Following are reasons which may cause the <b>bugs</b> : <ul style="list-style-type: none"> <li>Missing coding</li> <li>Wrong coding</li> </ul>             | The reason below leads to the <b>defects</b> : <ul style="list-style-type: none"> <li>Giving incorrect and wrong inputs.</li> <li>Dilemmas and</li> </ul>                                                                                                                                                         | The reasons for having an <b>error</b> are as follows: <ul style="list-style-type: none"> <li>Errors in the code.</li> <li>The Mistake of</li> </ul>                                                                                                                                   | The reasons behind the <b>fault</b> are as follows: <ul style="list-style-type: none"> <li>A Fault may occur by an improper step in the initial stage, process, or data</li> </ul>                                                                                           | Following are some of the most important reasons behind the <b>failure</b> : <ul style="list-style-type: none"> <li>Environmental condition</li> </ul> |

|                                   |                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                |                                                                                                                                                                                                                                     |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                   | Extra coding errors in the outside behavior and inside structure and design. An error in coding or logic affects the software and causes it to breakdown or the failure.  | some values. If a developer is unable to <b>compile or run a program successfully.</b> Confusions and issues in programming. Invalid login, loop, and syntax. Inconsistency between actual and expected outcomes. Blunders in design or requirement actions. Misperception in understanding the requirements of the application. | definition. Inconsistency or issue in the program. An irregularity or loophole in the software that leads the software to perform improperly.                                                                  | System usage<br>Users<br>Human error                                                                                                                                                                                                |
| <b>Way to prevent the reasons</b> | Following are the way to stop the <b>bugs:</b><br>Test-driven development. Offer programming language support. Adjusting, advanced, and operative development procedures. | With the help of the following, we can prevent the <b>Defects:</b><br>Implementing several innovative programming methods. Use of primary and correct software development techniques.                                                                                                                                           | Below are ways to prevent the <b>Errors:</b><br>Enhance the software quality with system review and programming. Detect the issues and prepare a suitable mitigation plan. Validate the fixes and verify their | The <b>fault</b> can be prevented with the help of the following:<br>Peer review. Assess the functional necessities of the software. Execute the detailed code analysis. Verify the correctness of software design and programming. |

|  |                                     |                                                                                              |                        |  |                             |
|--|-------------------------------------|----------------------------------------------------------------------------------------------|------------------------|--|-----------------------------|
|  | Evaluating the code systematically. | Peer review It is executing consistent code reviews to evaluate its quality and correctness. | quality and precision. |  | evaluate errors and issues. |
|--|-------------------------------------|----------------------------------------------------------------------------------------------|------------------------|--|-----------------------------|

## Software Testing Principles

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.

Let us see the seven different testing principles, one by one:

- Testing shows the presence of defects
- Exhaustive Testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context-dependent
- Absence of errors fallacy

### Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of



unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process.

### **Exhaustive Testing is not possible**

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

### **Early Testing**

Here early testing means that all the testing activities should start in the early stages of the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

### **Defect clustering**

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the **Pareto Principle**, which states that we can identify that approx. Eighty percent of the complication is present in 20

percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not be able to identify the new defects.

### **Pesticide paradox**

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

### **Testing is context-dependent**

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

### **Program Inspections**

Inspections are a formal type of review that involves checking the documents thoroughly before a meeting and is carried out mostly by moderators. A meeting is then held to review the code and the design. Inspection meetings can be held both physically and virtually.

The stages in the inspections process are: Planning, Overview meeting, Preparation, Inspection meeting, Rework and Follow-up. The Preparation, Inspection meeting and Rework stages might be iterated.

Code inspection in software engineering is the process of reviewing the code in an application to check for defects. Its purpose is to correct the issues in the programming language so the software performs at its highest potential.

Program inspections, also known as code inspections or peer reviews, are a systematic and structured approach to examining source code to identify defects, improve code quality, and ensure adherence to coding standards. Inspections involve a group of developers or peers reviewing the code, discussing potential issues, and suggesting improvements. The primary goal of program inspections is to catch errors early in the development process, before they manifest as problems in the running system.

Here are some key aspects of program inspections:

**Process:** Program inspections typically follow a well-defined process. The process involves planning and scheduling the inspection, selecting participants, conducting the inspection meeting, documenting the findings, and following up on the identified issues. The process may be guided by specific inspection guidelines or checklists to ensure consistent and thorough review.

**Participants:** Inspections involve a group of participants who review the code. The participants may include the code author, other developers, and peers with relevant expertise. The participation of multiple individuals with different perspectives helps in identifying diverse issues and promoting collective learning.

**Goals:** The primary goals of program inspections are to identify defects, improve code quality, and ensure adherence to coding standards. Defects can include logic errors, incorrect algorithms, poor error handling, or violations of coding conventions. Inspections also provide an opportunity to share knowledge, validate design decisions, and identify potential optimizations.

**Code Understanding:** Inspections require participants to thoroughly understand the code being reviewed. They need to comprehend the code's purpose, its interaction with other components, and its compliance with requirements and specifications. Code understanding is crucial for identifying potential issues and proposing effective solutions.

**Defect Identification:** During the inspection, participants actively search for defects or issues in the code. This may involve examining the code for syntax errors, logic flaws, performance bottlenecks, security vulnerabilities, or violations of coding best practices. The focus is on identifying defects that could impact the system's functionality, reliability, maintainability, or performance.

**Communication and Collaboration:** Inspections facilitate communication and collaboration among participants. Reviewers can discuss their findings, ask questions, and provide feedback to the code author. The inspection process encourages constructive criticism, open discussions, and knowledge sharing, which can lead to improved code quality and developer skill enhancement.

**Documentation and Follow-up:** The findings of the inspection, including identified

defects and suggested improvements, are documented. The code author is responsible for addressing the identified issues, either by making necessary code changes or providing justifications for not making changes. The inspection process may include a follow-up review to ensure that the identified issues have been appropriately resolved.

Program inspections complement other software development practices, such as testing and static analysis, by providing a human-focused, qualitative review of the code. They can help identify defects that may not be easily captured through automated techniques and contribute to overall code quality and reliability.

### **Stages of Testing: Unit Testing**

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.

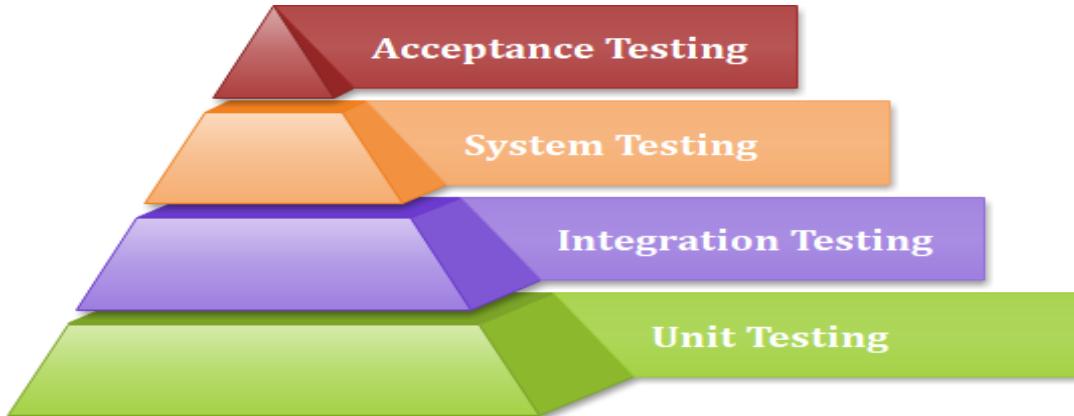
A unit is a single testable part of a software system and tested during the development phase of the application software.

The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.

Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as **Unit testing** or **components testing**.

### Why Unit Testing?

In a testing level hierarchy, unit testing is the first level of testing done before integration and other remaining levels of the testing. It uses modules for the testing process which reduces the dependency of waiting for Unit testing frameworks, stubs, drivers and mock objects are used for assistance in unit testing.



Generally, the software goes under four level of testing: Unit Testing, Integration Testing, System Testing, and Acceptance Testing but sometimes due to time consumption software testers does minimal unit testing but skipping of unit testing may lead to higher defects during Integration Testing, System Testing, and Acceptance Testing or even during Beta Testing which takes place after the completion of software application.

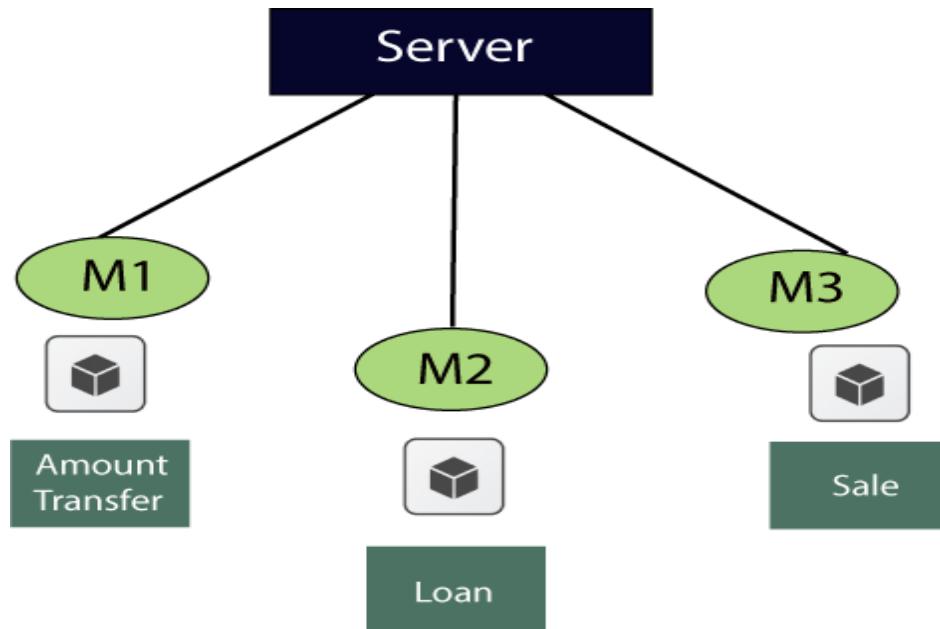
#### Some crucial reasons are listed below:

- Unit testing helps tester and developers to understand the base of code that makes them able to change defect causing code quickly.
- Unit testing helps in the documentation.

- Unit testing fixes defects very early in the development phase that's why there is a possibility to occur a smaller number of defects in upcoming testing levels.
- It helps with code reusability by migrating code and test cases.

### Example of Unit testing

Let us see one sample example for a better understanding of the concept of unit testing:



For the **amount transfer**, requirements are as follows:

|       |                                     |
|-------|-------------------------------------|
| 1.    | Amount transfer                     |
| 1.1   | From account number (FAN)→ Text Box |
| 1.1.1 | FAN→ accept only 4 digit            |
| 1.2   | To account no (TAN)→ Text Box       |
| 1.2.1 | TAN→ Accept only 4 digit            |

|       |                                 |
|-------|---------------------------------|
| 1.3   | Amount→ Text Box                |
| 1.3.1 | Amount → Accept maximum 4 digit |
| 1.4   | Transfer→ Button                |
| 1.4.1 | Transfer → Enabled              |
| 1.5   | Cancel→ Button                  |
| 1.5.1 | Cancel→ Enabled                 |

Below are the application access details, which is given by the customer

- URL→ login Page
- Username/password/OK → home page
- To reach Amount transfer module follow the below

**Loans → sales → Amount transfer**

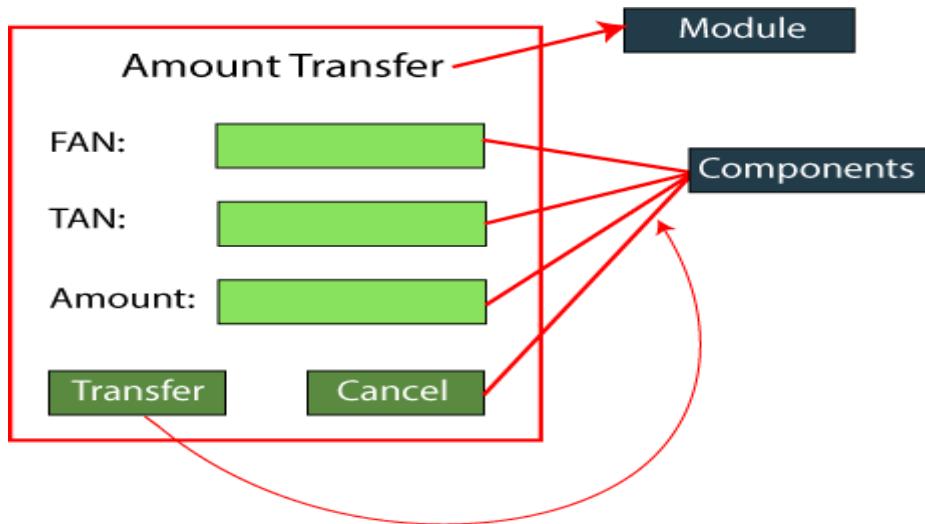
While performing unit testing, we should follow some rules, which are as follows:

- To start unit testing, at least we should have one module.
- Test for positive values
- Test for negative values
- No over testing
- No assumption required

When we feel that the **maximum test coverage** is achieved, we will stop the testing.

Now, we will start performing the unit testing on the different components such as

- **From account number(FAN)**
- **To account number(TAN)**



- **Amount**
- **Transfer**
- **Cancel**
- 

#### For the FAN components

| Values                   | Description                          |
|--------------------------|--------------------------------------|
| 1234                     | accept                               |
| 4311                     | Error message → account valid or not |
| blank                    | Error message → enter some values    |
| 5 digit/ 3 digit         | Error message → accept only 4 digit  |
| Alphanumeric             | Error message → accept only digit    |
| Blocked account no       | Error message                        |
| Copy and paste the value | Error message → type the value       |
| Same as FAN and TAN      | Error message                        |

**For the TAN component**

- Provide the values just like we did in **From account number** (FAN) components

**For Amount component**

- Provide the values just like we did in FAN and TAN components.

**For Transfer component**

- Enter valid FAN value
- Enter valid TAN value
- Enter the correct value of Amount
- Click on the Transfer button → amount transfer successfully( confirmation message)

**For Cancel Component**

- Enter the values of FAN, TAN, and amount.
- Click on the Cancel button → all data should be cleared.

## Unit Testing Tools

We have various types of unit testing tools available in the market, which are as follows:

- NUnit
- JUnit
- PHPunit
- Parasoft Jtest
- EMMA

When we have to find and authenticate the particular module or unit of the code, we need the unit testing tools. With the help of these tools, we can build secure design and documentation and decrease the bug count.

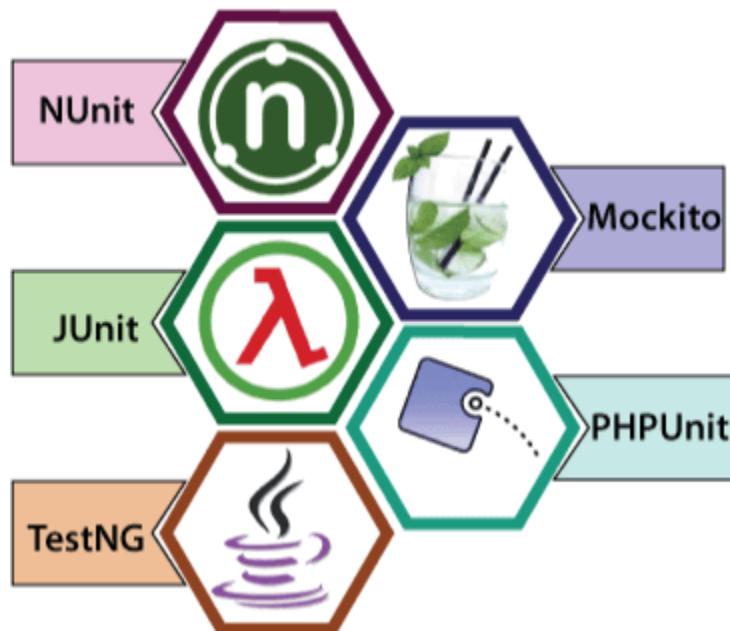
Generally, unit testing is a manual process, but now some of the organization has automated the unit test with the help of these tools. By using unit testing tools, we can cover the maximum coverage, performance, compatibility, and integration testing.

All the unit testing tool is implemented as a plug-in for the eclipse. Unit testing tools are used by the developers to test the source code of the application or to achieve the source code of the application.

Following are the most commonly used tools of [unit testing](#):

- **NUnit**
- **JUnit**
- **TestNG**
- **Mockito**
- **PHPUnit**

## Unit Testing Tools



### NUnit

One of the most commonly used unit testing tools is NUnit. It is an open-source tool and initially ported from the JUnit, which works for all .Net languages. NUnit was written entirely in the [C# language](#) and fully redesigned to get the advantage of many .Net language features. Like custom attributes and other reflection related capabilities.



### Features of NUnit

- It powerfully supports the data-driven tests.
- In this, we can execute the tests parallelly.
- It allows assertions as a static method of the asset class.
- It uses a console runner to load and execute tests.
- NUnit supports various platforms such as Silverlight, Xamarin mobile, .NET core, and compact framework.

### JUnit

It is another open-source unit testing framework, which was written in [Java programming language](#). It is mainly used in the development of the test-driven environment. Junit offers the annotation, which helps us to find the test method. This tool helps us to enhance the efficiency of the developer, which provides the consistency of the development code and reduces the time of the debugging.



### Feature of JUnit

- It offers the assertions for testing expected results.
- In this tool, we can quickly develop a code that enhances the quality of the code.
- This tool can be structured in the test suites, which have the test cases.
- To run the test, it gives the test runners.
- It will take less time to run the test cases.

## TestNG

It is an open-source tool, which supports Java and .Net programming languages. **Test Next Generation** (TestNG) is an advance unit testing tool, which is stimulated from JUnit and NUnit testing frameworks. Still, few new functionalities (Additional Annotation, Parallel Execution, Group Execution, Html Report, and Listener) make a TestNG more powerful tool.

For the automation process, TestNG will be used to handle the framework component and achieve the batch execution without any human interference.



## Feature of TestNG

Following are the some commonly used [features of TestNG](#):

- TestNG support various instance of the same test cases, parameterized, annotation, data-driven, functional, integration, and unit testing.
- In the case of development, TestNG will be used to develop a unit test case, and each unit test case will test the business logic of the source code.
- It will provide a flexible test configuration.
- It will have the Dependent methods for application server testing
- With the help of TestNG, we have full control over the test cases and the execution of the test cases.
- It is supported by multiple plug-in and tools such as IDEA, Eclipse, and Maven, and so on

## Mockito

It is a mocking framework that is used in the unit testing, and it was written in the Java programing language. Mockito is also an open-source tool introduced by the **MIT (Massachusetts Institute of Technology) License**.

With the help of Mockito, we can develop the testable application. The primary objective of using this tool is to simplify the development of a test by mocking external dependencies and use them in the test code. It can be used with other testing frameworks such as TestNG and Junit.



### Feature of Mockito

- It will be used to support exceptions.
- With the help of the annotation feature, we can produce the mocks.
- We do not need to write the mocks object on our own.
- It will support return values.
- It provides multiple methods like verify(), mock(), when(), etc., which are helpful to test Java applications.

### PHPUnit

Another unit testing tool is PHPUnit, which was written in PHP programming language. It is an instance of the xUnit architecture and based on the JUnit framework. It can generate the test results output in many various formats with [JSON](#), JUnit XML, TestDox, and Test anything protocol. We can run the test cases on the cross-platform operating system.



### Features of PHPUnit

- PHPUnit will provide the logging, code coverage analysis.
- Its development is hosted on GitHub.
- PHPUnit uses assertions to validate the performance of the specific component.

- With the help of this tool, developers can identify the issues in their newly developed code.

### Unit Testing Techniques:

Unit testing uses all white box testing techniques as it uses the code of software application:

- Data flow Testing
- Control Flow Testing
- Branch Coverage Testing
- Statement Coverage Testing
- Decision Coverage Testing

### How to achieve the best result via Unit testing?

Unit testing can give best results without getting confused and increase complexity by following the steps listed below:

- Test cases must be independent because if there is any change or enhancement in requirement, the test cases will not be affected.
- Naming conventions for unit test cases must be clear and consistent.
- During unit testing, the identified bugs must be fixed before jump on next phase of the SDLC.
- Only one code should be tested at one time.
- Adopt test cases with the writing of the code, if not doing so, the number of execution paths will be increased.
- If there are changes in the code of any module, ensure the corresponding unit test is available or not for that module.

### Advantages and disadvantages of unit testing

The pros and cons of unit testing are as follows:

### Advantages

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

### Disadvantages

- It cannot identify integration or broad level error as it works on units of the code.
- In the unit testing, evaluation of all execution paths is not possible, so unit testing is not able to catch each and every error in a program.
- It is best suitable for conjunction with other testing activities.

## Integration Testing

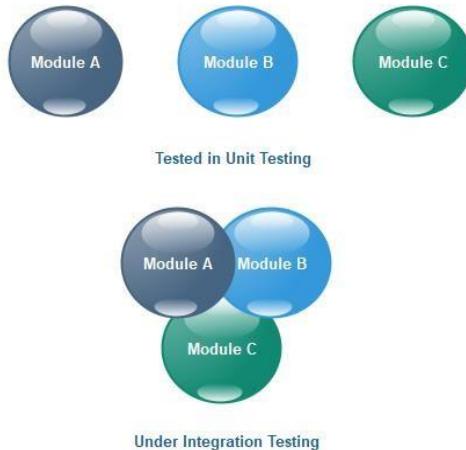
Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

**Unit testing** uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as **integration testing**

Let us see one sample example of a banking application, as we can see in the below image of amount transfer.

| Amount Transfer |                      |
|-----------------|----------------------|
| FAN:            | <input type="text"/> |
| TAN:            | <input type="text"/> |
| AMOUNT:         | <input type="text"/> |
| <b>Transfer</b> | <b>Cancel</b>        |



- First, we will login as a user **P** to amount transfer and send Rs200 amount, the confirmation message should be displayed on the screen as **amount transfer successfully**. Now logout as P and login as user **Q** and go to amount balance page and check for a balance in that account = Present balance + Received Balance. Therefore, the integration test is successful.
- Also, we check if the amount of balance has reduced by Rs200 in P user account.
- Click on the transaction, in P and Q, the message should be displayed regarding the data and time of the amount transfer.

### Guidelines for Integration Testing

- We go for the integration testing only after the functional testing is completed on each module of the application.
- We always do integration testing by picking module by module so that a proper sequence is followed, and also we don't miss out on any integration scenarios.
- First, determine the test case strategy through which executable test cases can be prepared according to test data.
- Examine the structure and architecture of the application and identify the crucial modules to test them first and also identify all possible scenarios.
- Design test cases to verify each interface in detail.

- Choose input data for test case execution. Input data plays a significant role in testing.
- If we find any bugs then communicate the bug reports to developers and fix defects and retest.
- Perform **positive and negative integration testing**.

Here **positive** testing implies that if the total balance is Rs15, 000 and we are transferring Rs1500 and checking if the amount transfer works fine. If it does, then the test would be a pass.

And **negative testing** means, if the total balance is Rs15, 000 and we are transferring Rs20, 000 and check if amount transfer occurs or not, if it does not occur, the test is a pass. If it happens, then there is a bug in the code, and we will send it to the development team for fixing that bug.

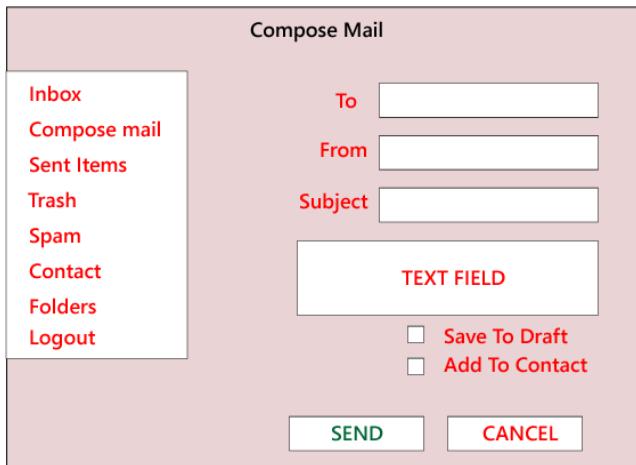
**For example:** In the Gmail application, the **Source** could be **Compose**, **Data** could be **Email** and the **Destination** could be the **Inbox**.

### Example of integration testing

Let us assume that we have a **Gmail** application where we perform the integration testing.

First, we will do **functional testing** on the **login page**, which includes the various components such as **username**, **password**, **submit**, and **cancel** button. Then only we can perform integration testing.

The different integration scenarios are as follows:



### Scenarios1:

- First, we login as **P** users and click on the **Compose** mail and performing the functional testing for the specific components.
- Now we click on the **Send** and also check for **Save Drafts**.
- After that, we send a **mail** to **Q** and verify in the **Sent Items** folder of **P** to check if the send mail is there.
- Now, we will **log out** as **P** and login as **Q** and move to the **Inbox** and verify that if the mail has reached.

**Secanrios2:** We also perform the integration testing on **Spam** folders. If the particular contact has been marked as spam, then any mail sent by that user should go to the spam folder and not in the inbox.

As we can see in the below image, we will perform the functional testing for all the **text fields and every feature**. Then we will perform **integration testing** for the related functions. We first test the **add user**, **list of users**, **delete user**, **edit user**, and then **search user**.

**Add Users**

|                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                           |                                                                                                                                                                                                                                                                                                        |                                  |                                              |                |                                                          |                                                                                                                    |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|----------------------------------------------|----------------|----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|--|
| <b>Add User</b><br><b>Delete User</b><br><b>List User</b><br><b>Edit User</b><br><b>Product Sales</b><br><b>Product Purchases</b><br><b>Search Users</b><br><b>Help</b> | <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%; vertical-align: top; padding-right: 10px;"> <b>User Name</b><br/> <b>Password</b><br/> <b>Designation</b> </td> <td style="width: 70%; text-align: right; vertical-align: bottom;"> <input type="text"/><br/> <input type="text"/><br/> <input style="width: 150px; height: 20px;" type="text"/> <div style="margin-top: 5px; text-align: right;"> <span style="color: red;">Team Lead</span><br/> <span style="color: red;">Manager</span><br/> <span style="color: red;">.....</span> </div> </td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;"> <b>Email</b><br/> <b>Telephone</b> </td> <td style="text-align: right; vertical-align: bottom;"> <input type="text"/><br/> <input type="text"/> </td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;"> <b>Address</b> </td> <td style="text-align: right; vertical-align: bottom;"> <input style="width: 150px; height: 40px;" type="text"/> </td> </tr> <tr> <td colspan="2" style="text-align: right; padding-top: 10px;"> <input type="button" value="Submit"/> <input style="color: red; margin-left: 10px;" type="button" value="Cancel"/> </td> </tr> </table> | <b>User Name</b><br><b>Password</b><br><b>Designation</b> | <input type="text"/><br><input type="text"/><br><input style="width: 150px; height: 20px;" type="text"/> <div style="margin-top: 5px; text-align: right;"> <span style="color: red;">Team Lead</span><br/> <span style="color: red;">Manager</span><br/> <span style="color: red;">.....</span> </div> | <b>Email</b><br><b>Telephone</b> | <input type="text"/><br><input type="text"/> | <b>Address</b> | <input style="width: 150px; height: 40px;" type="text"/> | <input type="button" value="Submit"/> <input style="color: red; margin-left: 10px;" type="button" value="Cancel"/> |  |
| <b>User Name</b><br><b>Password</b><br><b>Designation</b>                                                                                                               | <input type="text"/><br><input type="text"/><br><input style="width: 150px; height: 20px;" type="text"/> <div style="margin-top: 5px; text-align: right;"> <span style="color: red;">Team Lead</span><br/> <span style="color: red;">Manager</span><br/> <span style="color: red;">.....</span> </div>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                           |                                                                                                                                                                                                                                                                                                        |                                  |                                              |                |                                                          |                                                                                                                    |  |
| <b>Email</b><br><b>Telephone</b>                                                                                                                                        | <input type="text"/><br><input type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                           |                                                                                                                                                                                                                                                                                                        |                                  |                                              |                |                                                          |                                                                                                                    |  |
| <b>Address</b>                                                                                                                                                          | <input style="width: 150px; height: 40px;" type="text"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                           |                                                                                                                                                                                                                                                                                                        |                                  |                                              |                |                                                          |                                                                                                                    |  |
| <input type="button" value="Submit"/> <input style="color: red; margin-left: 10px;" type="button" value="Cancel"/>                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                           |                                                                                                                                                                                                                                                                                                        |                                  |                                              |                |                                                          |                                                                                                                    |  |

**Note:**

- There are some features, we might be performing only the **functional testing**, and there are some features where we are performing both **functional** and **integration testing** based on the feature's requirements.
- **Prioritizing is essential**, and we should perform it at all the phases, which means we will open the application and select which feature needs to be tested first. Then go to that feature and choose which component must be tested first. Go to those components and determine what values to be entered first. And don't apply the same rule everywhere because testing logic varies from feature to feature.
- While performing testing, we should test one feature entirely and then only proceed to another function.
- Among the two features, we must be performing **only positive integrating testing** or both **positive and negative integration** testing, and this also depends on the features need.

**Reason Behind Integration Testing**

Although all modules of software application already tested in unit testing, errors still exist due to the following reasons:

1. Each module is designed by individual software developer whose programming logic may differ from developers of other modules so; integration testing becomes essential to determine the working of software modules.
2. To check the interaction of software modules with the database whether it is an erroneous or not.
3. Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing becomes mandatory.
4. Incompatibility between modules of software could create errors.
5. To test hardware's compatibility with software.
6. If exception handling is inadequate between modules, it can create bugs.

### Integration Testing Techniques

Any testing technique (Blackbox, Whitebox, and Greybox) can be used for Integration Testing; some are listed below:

#### Black Box Testing

- State Transition technique
- Decision Table Technique
- Boundary Value Analysis
- All-pairs Testing
- Cause and Effect Graph
- Equivalence Partitioning
- Error Guessing

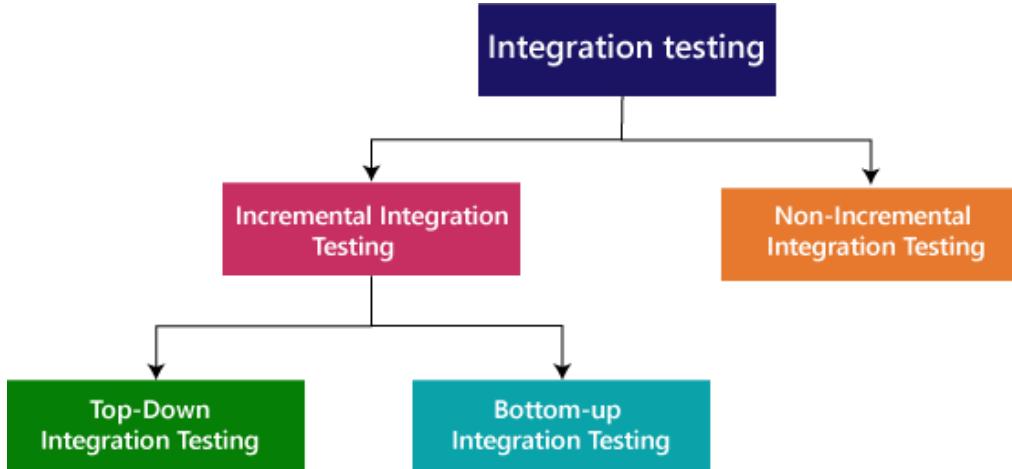
#### White Box Testing

- Data flow testing
- Control Flow Testing
- Branch Coverage Testing
- Decision Coverage Testing

## Types of Integration Testing

Integration testing can be classified into two parts:

- **Incremental integration testing**
- **Non incremental integration testing**

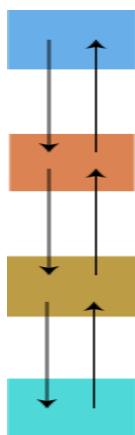


### Incremental Approach

In the Incremental Approach, modules are added in ascending order one by one or according to need. The selected modules must be logically related. Generally, two or more than two modules are added and tested to determine the correctness of functions. The process continues until the successful testing of all the modules.

OR

In this type of testing, there is a strong relationship between the dependent modules. Suppose we take two or more modules and verify that the data flow between them is working fine. If it is, then add more modules and test again.



**For example:** Suppose we have a Flipkart application, we will perform incremental integration testing, and the flow of the application would like this:

Flipkart → Login → Home → Search → Add cart → Payment → Logout

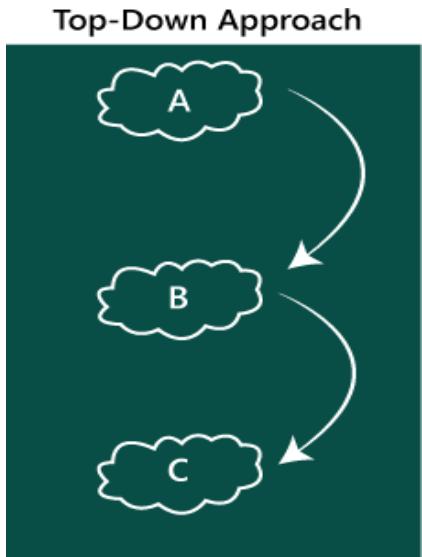
Incremental integration testing is carried out by further methods:

- Top-Down approach
- Bottom-Up approach

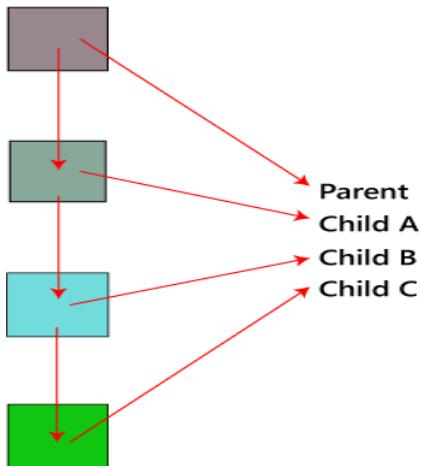
### Top-Down Approach

The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules

tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.



In the top-down approach, we will be ensuring that the module we are adding is the **child of the previous one like Child C is a child of Child B** and so on as we can see in the below image:



#### Advantages:

- Identification of defect is difficult.
- An early prototype is possible.

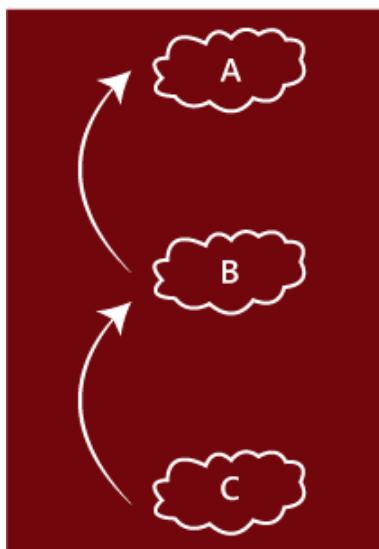
#### Disadvantages:

- Due to the high number of stubs, it gets quite complicated.
- Lower level modules are tested inadequately.
- Critical Modules are tested first so that fewer chances of defects.

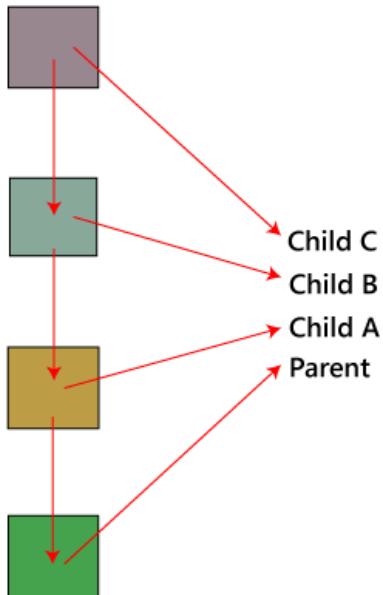
### Bottom-Up Method

The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from **bottom to the top** and check the data flow in the same order.

#### Bottom-up Approach



In the bottom-up method, we will ensure that the modules we are adding **are the parent of the previous one** as we can see in the below image:



## Advantages

- Identification of defect is easy.
- Do not need to wait for the development of all the modules as it saves time.

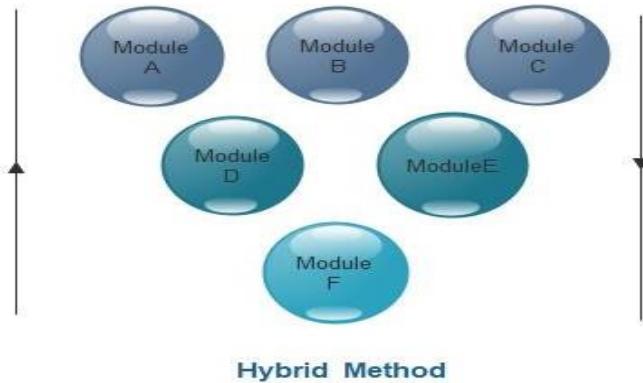
## Disadvantages

- Critical modules are tested last due to which the defects can occur.
- There is no possibility of an early prototype.

In this, we have one addition approach which is known as **hybrid testing**.

## Hybrid Testing Method

In this approach, both **Top-Down** and **Bottom-Up** approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously. There is less possibility of occurrence of defect because each module interface is tested.



### Advantages

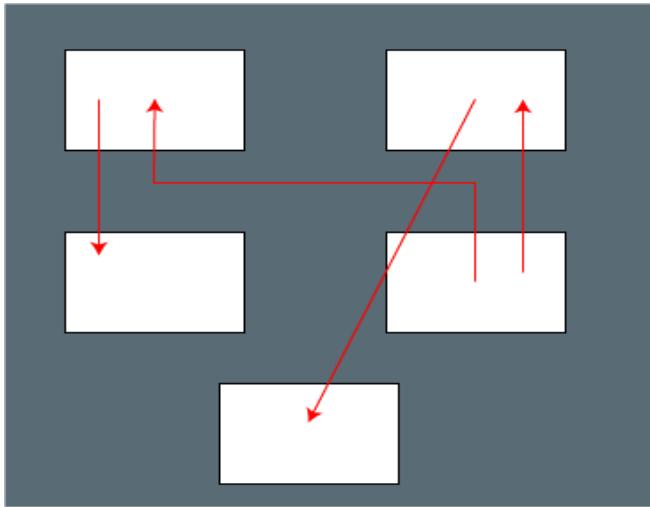
- The hybrid method provides features of both Bottom Up and Top Down methods.
- It is most time reducing method.
- It provides complete testing of all modules.

### Disadvantages

- This method needs a higher level of concentration as the process carried out in both directions simultaneously.
- Complicated method.

### Non-incremental integration testing

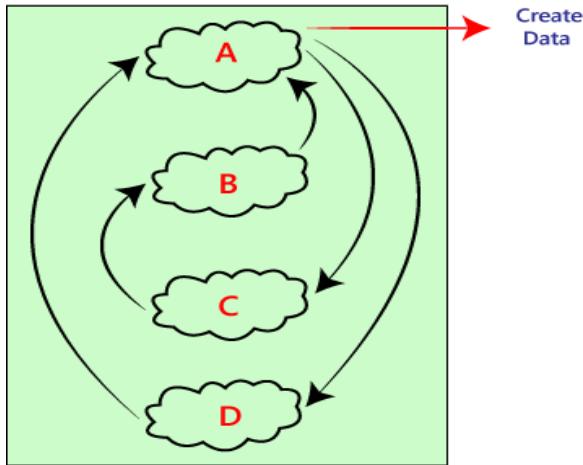
We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the **Big bang method**.



### Big Bang Method

In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.



### Advantages:

- It is convenient for small size software systems.

**Disadvantages:**

- Identification of defects is difficult because finding the error where it came from is a problem, and we don't know the source of the bug.
- Small modules missed easily.
- Time provided for testing is very less.
- We may miss to test some of the interfaces.

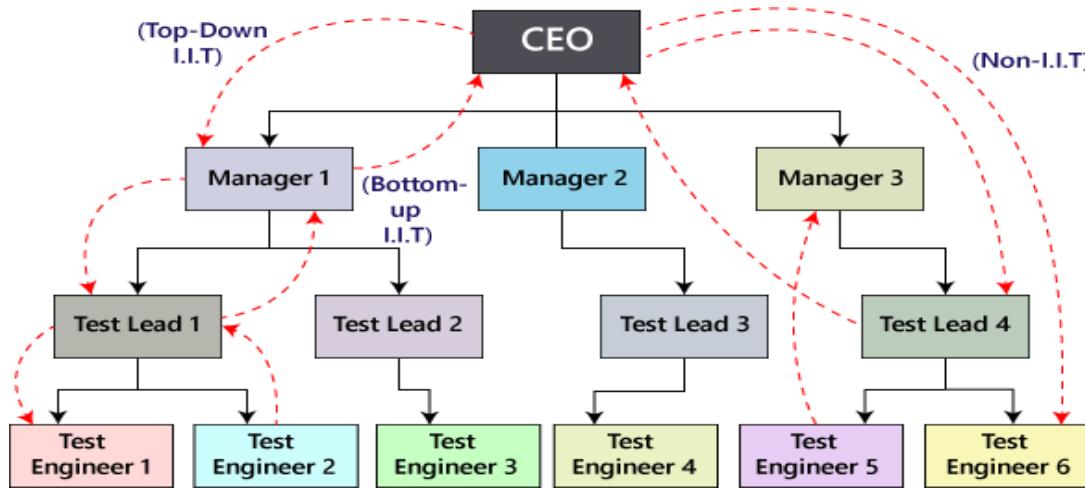
Let us see examples for our better understanding of the non-incremental integrating testing or big bang method:

**Example1**

In the below example, the development team develops the application and sends it to the CEO of the testing team. Then the CEO will log in to the application and generate the username and password and send a mail to the manager. After that, the CEO will tell them to start testing the application.

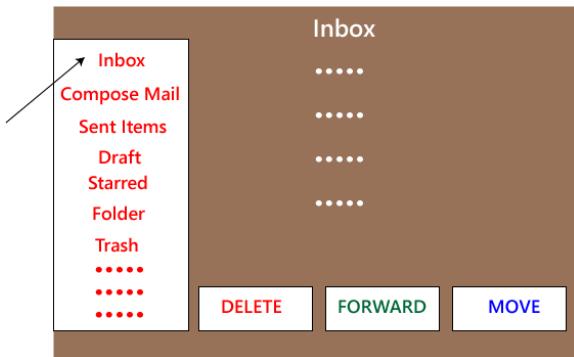
Then the manager manages the username and the password and produces a username and password and sends it to the **test leads**. And the **test leads** will send it to the **test engineers** for further testing purposes. This order from the CEO to the test engineer is **top-down incremental integrating testing**.

In the same way, when the test engineers are done with testing, they send a report to the **test leads**, who then submit a report to the **manager**, and the manager will send a report to the **CEO**. This process is known as **Bottom-up incremental integration testing** as we can see in the below image:



## Example2

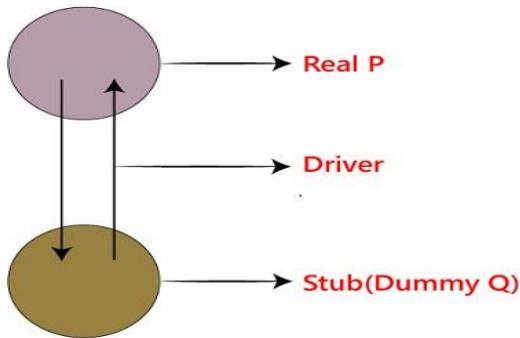
The below example demonstrates a home page of **Gmail's Inbox**, where we click on the **Inbox** link, and we are moved to the inbox page. Here we have to do **non-incremental integration testing** because there is no parent and child concept.



## Note

### Stub and driver

The **stub** is a dummy module that receives the data and creates lots of probable data, but it performs like a real module. When a data is sent from module P to Stub Q, it receives the data without confirming and validating it, and produce the estimated outcome for the given data.



The function of a driver is used to verify the data from P and sends it to stub and also checks the expected data from the stub and sends it to P.

The **driver** is one that sets up the test environments and also takes care of the communication, evaluates results, and sends the reports. We never use the stub and driver in the testing process.

In **White box testing**, bottom-up integration testing is ideal because writing drivers is accessible. And in **black box testing**, no preference is given to any testing as it depends on the application.

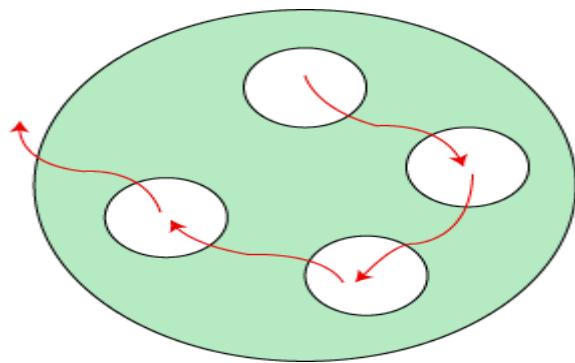
## System Testing

**System Testing** includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different type of tests with the purpose to exercise and examine the full working of an integrated software computer system against requirements.



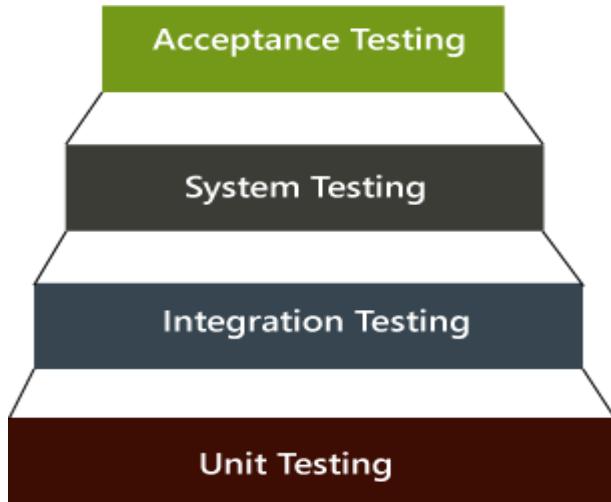
To check the end-to-end flow of an application or the software as a user is known as **System testing**. In this, we navigate (go through) all the necessary modules of an application and check if the end features or the end business works fine, and test the product as a whole system.

It is **end-to-end testing** where the testing environment is similar to the production environment.



There are four levels of **software testing**: **unit testing**, **integration testing**, **system testing** and **acceptance testing**, all are used for the testing purpose. Unit Testing used to test a single software; Integration Testing used to test a group of units of software, System Testing used to test a whole system and Acceptance Testing used to test the acceptability of business requirements. Here we are discussing system testing which is the third level of testing levels

#### Hierarchy of Testing Levels



There are mainly two widely used methods for software testing, one is **White box testing** which uses internal coding to design test cases and another is **black box testing** which uses GUI or user perspective to develop test cases.

- White box testing
- Black box testing

**System testing falls under Black box testing** as it includes testing of the external working of the software. Testing follows user's perspective to identify minor defects.

System Testing includes the following steps.

- Verification of input functions of the application to test whether it is producing the expected output or not.
- Testing of integrated software by including external peripherals to check the interaction of various components with each other.
- Testing of the whole system for End to End testing.
- Behavior testing of the application via a user's experience

### Example of System testing

Suppose we open an application, let say [www.rediff.com](http://www.rediff.com), and there we can see that an advertisement is displayed on the top of the homepage, and it remains there for a

few seconds before it disappears. These types of Ads are done by the Advertisement Management System (AMS). Now, we will perform system testing for this type of field.

The below application works in the following manner:

- Let's say that Amazon wants to display a promotion ad on January 26 at precisely 10:00 AM on the Rediff's home page for the country India.
- Then, the sales manager logs into the website and creates a request for an advertisement dated for the above day.
- He/she attaches a file that likely an image files or the video file of the AD and applies.
- The next day, the AMS manager of Rediffmail login into the application and verifies the awaiting Ad request.
- The AMS manager will check those Amazons ad requests are pending, and then he/she will check if the space is available for the particular date and time.
- If space is there, then he/she evaluate the cost of putting up the Ad at 15\$ per second, and the overall Ad cost for 10 seconds is approximate 150\$.
- The AMS manager clicks on the payment request and sends the estimated value along with the request for payment to the Amazon manager.
- Then the amazon manager login into the Ad status and confirms the payment request, and he/she makes the payment as per all the details and clicks on the **Submit and Pay**
- As soon as Rediff's AMs manager gets the amount, he/she will set up the Advertisement for the specific date and time on the Rediffmail's home page.

The various system test scenarios are as follows:

**Scenario1:** The first test is the general scenario, as we discussed above. The test engineer will do the system testing for the underlying situation where the Amazon manager creates a request for the Ad and that Ad is used at a particular date and time.

**Scenario2:** Suppose the Amazon manager feels that the AD space is too expensive and cancels the request. At the same time, the Flipkart requests the Ad space on January 26 at 10:00 AM. Then the request of Amazon has been canceled. Therefore, Flipkart's promotion ad must be arranged on January 26 at 10 AM.

After all, the request and payment have been made. Now, if Amazon changes their mind and they feel that they are ready to make payment for January 26 at 10 AM, which should be given because Flipkart has already used that space. Hence, another calendar must open up for Amazon to make their booking.

**Scenario3:** in this, first, we login as AMS manger, then click on Set Price page and set the price for AD space on logout page to 10\$ per second.

Then login as Amazon manager and select the date and time to put up and Ad on the logout page. And the payment should be 100\$ for 10 seconds of an Ad on Rediffmail logout page.

| Name                                  | Status                                | Total Amount         |
|---------------------------------------|---------------------------------------|----------------------|
| Amazon                                | Available                             | 150\$                |
| Flipkart                              | Available                             | 100\$                |
| ....                                  | ....                                  | ....                 |
| <input type="button" value="Pay"/>    | <input type="button" value="Cancel"/> |                      |
| Credit Card No.                       |                                       | <input type="text"/> |
| ....                                  | ....                                  | ....                 |
| <input type="button" value="SUBMIT"/> | <input type="button" value="CANCEL"/> |                      |

| REQUEST FOR ADVERTISEMENT                                                    |                                                                     |
|------------------------------------------------------------------------------|---------------------------------------------------------------------|
| Product                                                                      | <input type="text" value="Amazon"/>                                 |
| Country                                                                      | <input type="text" value="India"/> <input type="button" value="▼"/> |
|                                                                              | China<br>Russia<br>U.S.A                                            |
| Page                                                                         | <input type="text" value="Inbox"/> <input type="button" value="▼"/> |
|                                                                              | Sent Items<br>Compose Mail<br>Logout                                |
| Duration                                                                     | <input type="text" value="10 Second"/>                              |
| Date                                                                         | <input type="text" value="26th January 10:00AM"/>                   |
| <input type="button" value="Attach Advertisement [Text,Audio,Image,Video]"/> |                                                                     |
| <input type="button" value="SUBMIT"/>                                        | <input type="button" value="CANCEL"/>                               |

| PENDING ADVERTISEMENT                             |                                                |                          |
|---------------------------------------------------|------------------------------------------------|--------------------------|
| NAME                                              | STATUS                                         | SELECT                   |
| ....                                              | ....                                           | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| Pending Ads                                       | ....                                           | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| Set Rates                                         | ....                                           | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| Amazon                                            | Pending                                        | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| Flipkart                                          | Pending                                        | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| ....                                              | ....                                           | <input type="checkbox"/> |
| <input type="button" value="CHECK AVAILABILITY"/> | <input type="button" value="PAYMENT REQUEST"/> |                          |
| <input type="button" value="APPROXIMATE"/>        | <input type="button" value="SETUP"/>           |                          |

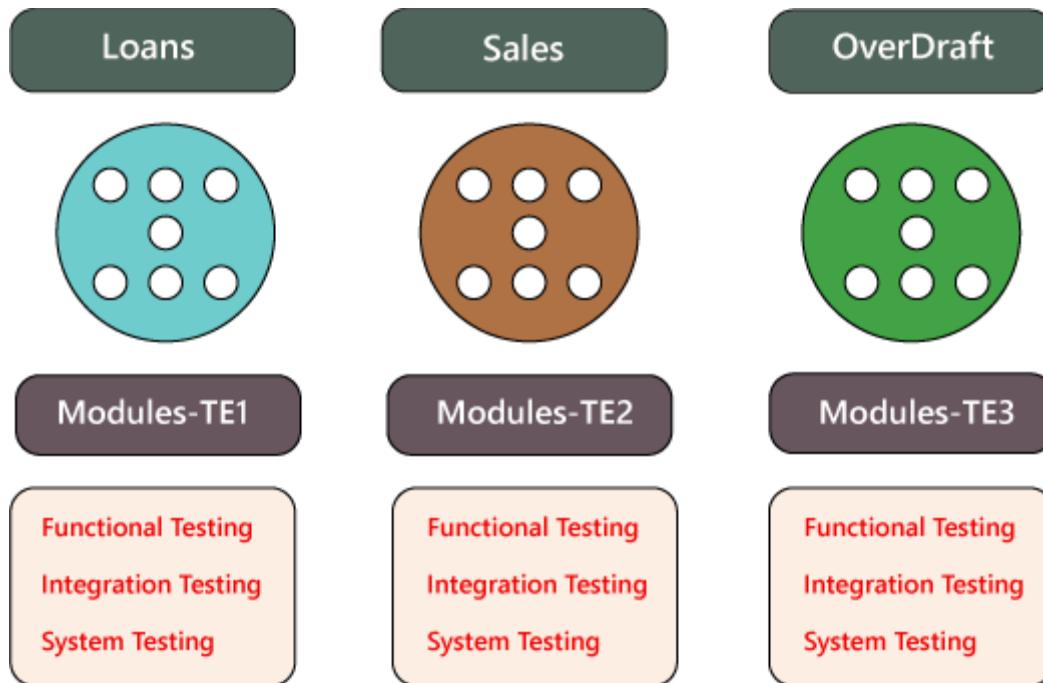
| SET PRICE                             |                                                                      |
|---------------------------------------|----------------------------------------------------------------------|
| Country                               | <input type="text" value="India"/> <input type="button" value="▼"/>  |
| Page                                  | <input type="text" value="Logout"/> <input type="button" value="▼"/> |
| Duration                              | <input type="text" value="10 Seconds"/>                              |
| Amount                                | <input type="text" value="10\$"/>                                    |
| <input type="button" value="SUBMIT"/> | <input type="button" value="CANCEL"/>                                |

Amazon Advertisement Available at 10:00AM on January 26th

As we can see in the below image, we have three different modules like **Loans**, **Sales**, and **Overdraft**. And these modules are going to be tested by their assigned test engineers only because if data flow between these modules or scenarios, then we need to clear that in which module it is going and that test engineer should check that thing.

Let us assume that here we are performing system testing on the interest estimation, where the customer takes the Overdraft for the first time as well as for the second time



## Scenario 2

Now, we test the alternative scenario where the bank provides an offer, which says that a customer who takes Rs45000 as Overdraft for the first time will not charge for the Process fee. The processing fee will not be refunded when the customer chooses another overdraft for the third time.

We have to test for the third scenario, where the customer takes the Overdraft of Rs45000 for the first time, and also verify that the Overdraft repays balance after applying for another overdraft for the third time.

## Scenario 3

In this, we will reflect that the application is being used generally by all the clients, all of a sudden the bank decided to reduce the processing fee to Rs100 for new customer, and we have test Overdraft for new clients and check whether it is accepting only for Rs100.

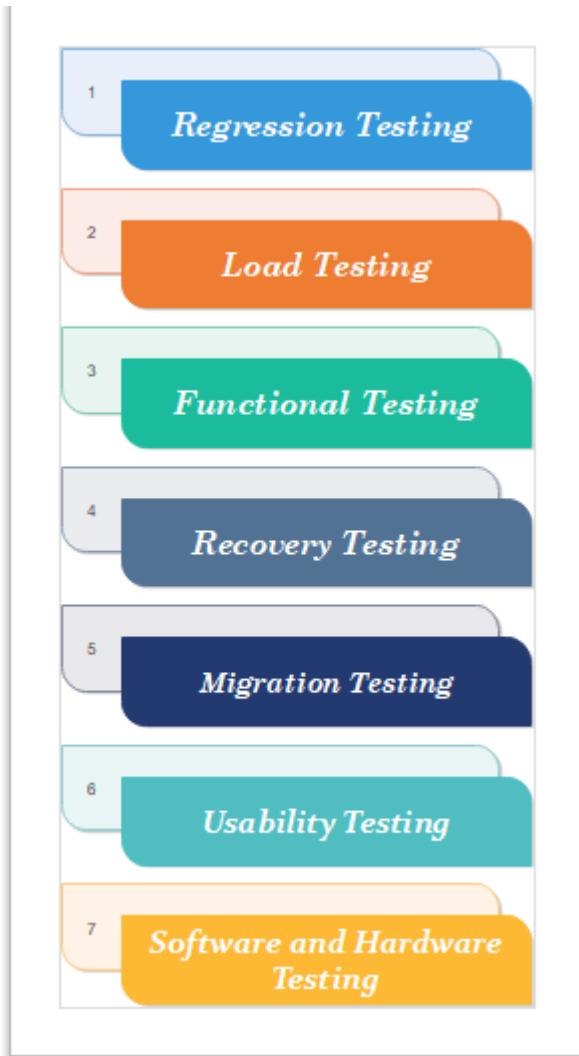
But then we get conflicts in the requirement, assume the client has applied for Rs15000 as Overdraft with the current process fee for Rs200. Before the Manager is yet to approve it, the bank decreases the process fee to Rs100.

Now, we have to test what process fee is charged for the Overdraft for the pending customer. And the testing team cannot assume anything; they need to communicate with the Business Analyst or the Client and find out what they want in those cases.

Therefore, if the customers provide the first set of requirements, we must come up with the maximum possible scenarios.

### Types of System Testing

System testing is divided into more than 50 types, but software testing companies typically uses some of them. These are listed below:



#### Regression Testing

Regression testing is performed under system testing to confirm and identify that if there's any defect in the system due to modification in any other part of the system. It makes sure, any changes done during the development process have not introduced a new defect and also gives assurance; old defects will not exist on the addition of new software over the time.

## Load Testing

Load testing is performed under system testing to clarify whether the system can work under real-time loads or not.

## Functional Testing

Functional testing of a system is performed to find if there's any missing function in the system. Tester makes a list of vital functions that should be in the system and can be added during functional testing and should improve quality of the system.

## Recovery Testing

Recovery testing of a system is performed under system testing to confirm reliability, trustworthiness, accountability of the system and all are lying on recouping skills of the system. It should be able to recover from all the possible system crashes successfully.

In this testing, we will test the application to check how well it recovers from the crashes or disasters.

Recovery testing contains the following steps:

- Whenever the software crashes, it should not vanish but should write the **crash log message or the error log message** where the reason for crash should be mentioned. **For example: C://Program Files/QTP/Cresh.log**
- It should kill its own procedure before it vanishes. Like, in Windows, we have the Task Manager to show which process is running.
- We will introduce the bug and crash the application, which means that someone will lead us to how and when will the application crash. Or **By experiences**, after few months of involvement on working the product, we can get to know how and when the application will crash.
- Re-open the application; the application must be reopened with earlier settings.

**For example:** Suppose, we are using the Google Chrome browser, if the power goes off, then we switch on the system and re-open the Google chrome, we get a message asking whether we want to **start a new session or restore the previous session**. For any developed product, the developer writes a recovery program that describes, why the software or the application is crashing, whether the crash log messages are written or not, etc.

## Migration Testing

Migration testing is performed to ensure that if the system needs to be modified in new infrastructure so it should be modified without any issue.

## Usability Testing

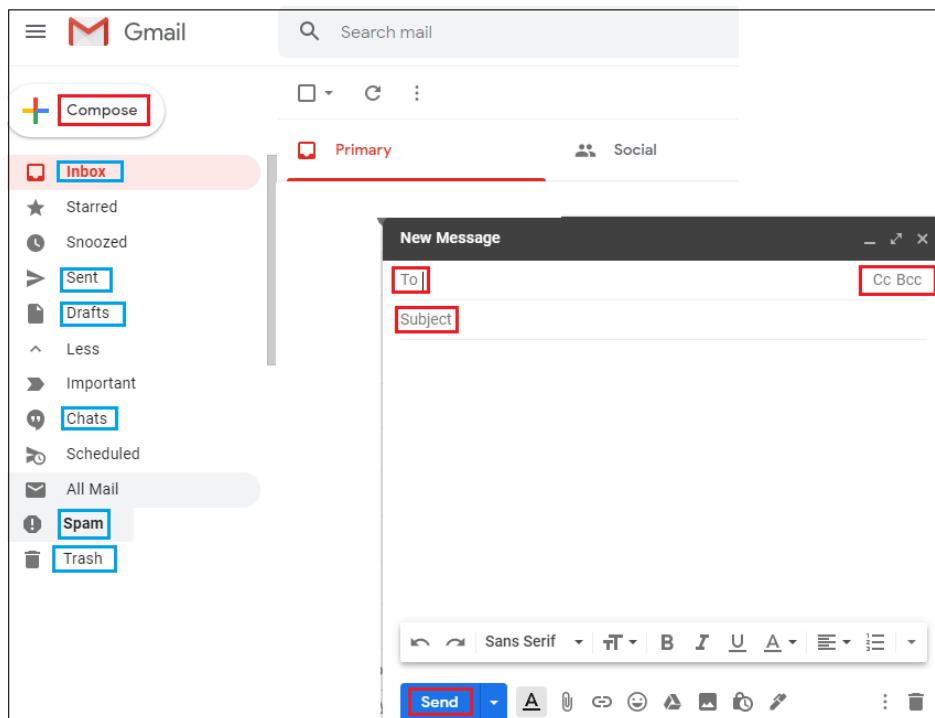
The purpose of this testing to make sure that the system is well familiar with the user and it meets its objective for what it supposed to do.

## Software and Hardware Testing

This testing of the system intends to check hardware and software compatibility. The hardware configuration must be compatible with the software to run it without any issue. Compatibility provides flexibility by providing interactions between hardware and software.

## Why is System Testing Important?

- System Testing gives hundred percent assurance of system performance as it covers end to end function of the system.
- It includes testing of System software architecture and business requirements.
- It helps in mitigating live issues and bugs even after production.
- System testing uses both existing system and a new system to feed same data in both and then compare the differences in functionalities of added and existing functions so, the user can understand benefits of new added functions of the system.



### Testing Any Application

Here, we are going to test the **Gmail** application to understand how **functional**, **integration**, and **System testing** works.

Suppose, we have to test the various modules such as **Login, Compose, Draft, Inbox, Sent Item, Spam, Chat, Help, Logout** of Gmail application.

We do **Functional Testing** on all Modules First, and then only we can perform integration testing and system testing.

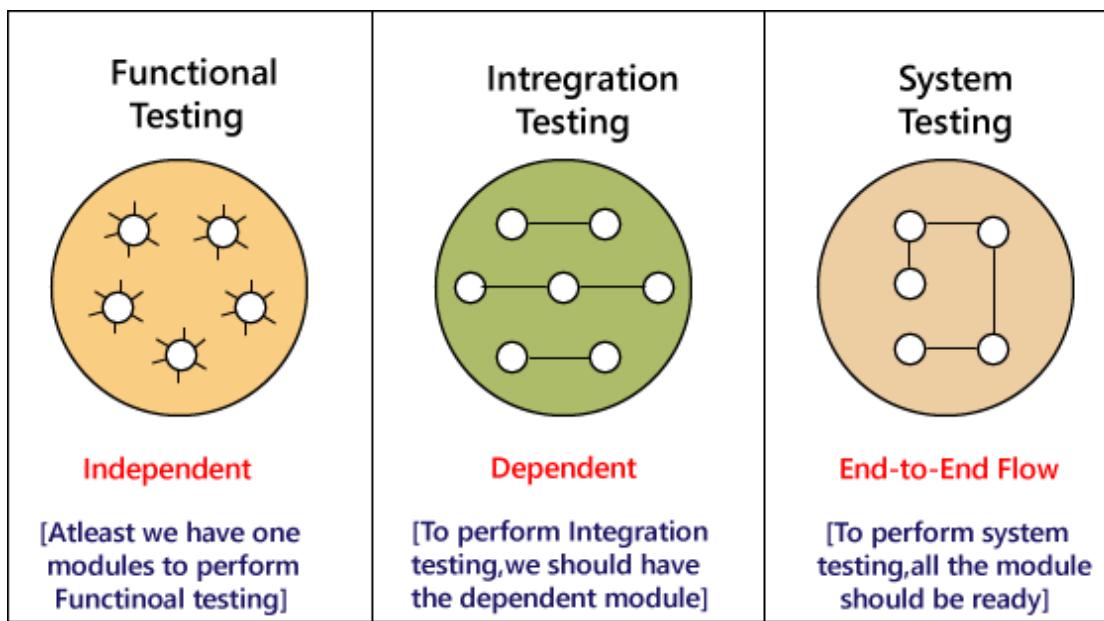
In functional testing, at least we have one module to perform functional testing. So here we have the Compose Module where we are performing the functional testing.

### Compose

The different components of the Compose module are **To, CC, BCC, Subject, Attachment, Body, Sent, Save to Draft, Close**.

- First, we will do functional testing on the **To**

| Input                  | Results |
|------------------------|---------|
| <b>Positive inputs</b> |         |
| mike@gmail.com         | Accept  |



|                        |        |
|------------------------|--------|
| Mike12@gmail.com       | Accept |
| Mike@yahoo.com         | Accept |
| <b>Negative inputs</b> |        |
| Mike@yahoocom          | Error  |
| Mike@yahoo.com         | Error  |

- o For **CC & BCC** components, we will take the same input as **To component**.
- o For **Subject** component, we will take the following inputs and scenarios:

| Input                   | Results |
|-------------------------|---------|
| <b>Positive inputs</b>  |         |
| Enter maximum character | Accept  |
| Enter Minimum character | Accept  |
| Blank Space             | Accept  |
| URL                     | Accept  |
| Copy & Paste            | Accept  |
| <b>Negative inputs</b>  |         |
| Crossed maximum digits  | Error   |

|                              |       |
|------------------------------|-------|
| Paste images / video / audio | Error |
|------------------------------|-------|

- **Maximum character**
- **Minimum character**
- **Flash files (GIF)**
- **Smiles**
- **Format**
- **Blank**
- **Copy & Paste**
- **Hyperlink**
- **Signature**
  
- For the **Attachment** component, we will take the help of the below scenarios and test the component.
  - **File size at maximum**
  - **Different file formats**
  - **Total No. of files**
  - **Attach multiple files at the same time**
  - **Drag & Drop**
  - **No Attachment**
  - **Delete Attachment**
  - **Cancel Uploading**
  - **View Attachment**
  - **Browser different locations**
  - **Attach opened files**
  
- For **Sent** component, we will write the entire field and click on the **Sent** button, and the Confirmation message; **Message sent successfully** must be displayed.
- For **Saved to Drafts** component, we will write the entire field and click on **aved to drafts**, and the Confirmation message must be displayed.

- For the **Cancel** component, we will write all fields and click on the Cancel button, and the **Window will be closed** or moved to **save to draft** or all fields must be refreshed.

Once we are done performing functional testing on compose module, we will do the Integration testing on Gmail application's various modules:

### **Login**

- First, we will enter the username and password for login to the application and Check the username on the Homepage.

### **Compose**

- Compose mail, send it and check the mail in Sent Item [sender]
- Compose mail, send it and check the mail in the receiver [Inbox]
- Compose mail, send it and check the mail in self [Inbox]
- Compose mail, click on Save as Draft, and check-in sender draft.
- Compose mail, send it invalid id (valid format), and check for undelivered message.
- Compose mail, close and check-in Drafts.

### **Inbox**

- Select the mail, reply, and check in sent items or receiver Inbox.
- Select the mail in Inbox for reply, Save as Draft and check in the Draft.
- Select the mail then delete it, and check in Trash.

### **Sent Item**

- Select the mail, Sent Item, Reply or Forward, and check in Sent item or receiver inbox.
- Select mail, Sent Item, Reply or Forward, Save as Draft, and verify in the Draft.
- Select mail, delete it, and check in the Trash.

### **Draft**

- Select the email draft, forward and check Sent item or Inbox.
- Select the email draft, delete and verify in Trash.

**Chat**

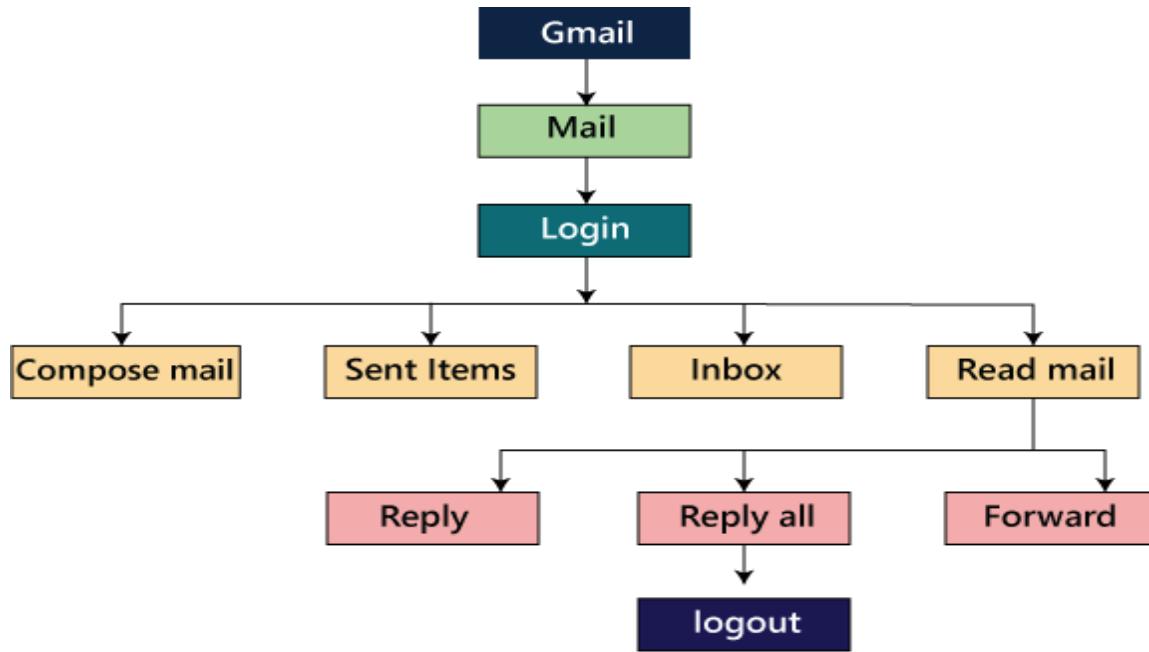
- Chat with offline users saved in the inbox of the receiver.
- Chat with the user and verify it in the chat window.
- Chat with a user and check in the chat history.

**UNIT II TEST PLANNING****6**

The Goal of Test Planning, High Level Expectations, Intergroup Responsibilities, Test Phases, Test Strategy, Resource Requirements, Tester Assignments, Test Schedule, Test Cases, Bug Reporting, Metrics and Statistics.

**The Goal of Test Planning**

A test plan is a detailed document which describes software testing areas and activities. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables.



The goal of test planning is to define a comprehensive strategy and approach for testing a software application or system. It involves determining the scope of testing, identifying testing objectives, defining test objectives, identifying necessary resources, and creating a schedule for executing the tests.

The main objectives of test planning are as follows:

**Understanding Testing Scope:** Test planning helps in defining the scope of testing, including the features and functionalities to be tested. It involves analyzing requirements, design specifications, and other project documents to identify what needs to be tested.

**Defining Testing Objectives:** Test planning involves setting clear and measurable testing objectives. These objectives may include validating system functionality, identifying defects, assessing performance and scalability, verifying security measures, and ensuring regulatory compliance, among others.

**Identifying Test Techniques and Methods:** Test planning involves selecting appropriate testing techniques and methods to achieve the testing objectives. This includes determining whether manual or automated testing is suitable, deciding on the types of tests to be conducted (e.g., functional, performance, security), and identifying any specific tools or frameworks that may be required.

**Allocating Testing Resources:** Test planning helps in identifying the necessary resources for testing, including personnel, hardware, software, and testing environments. It involves determining the skills and

**Creating Test Schedule:** Test planning involves creating a timeline or schedule for executing the tests. It includes defining milestones, setting deadlines, and establishing a sequence of testing activities. The test schedule should consider dependencies on other project activities and take into account any constraints or limitations.

**Risk Assessment and Mitigation:** Test planning involves identifying and assessing potential risks and uncertainties associated with testing. This includes analyzing factors that could impact the success of testing, such as technical challenges, resource constraints, time limitations, and external dependencies. Risk mitigation strategies and contingency plans are developed to address these potential risks.

**Documentation and Communication:** Test planning involves documenting the test strategy, test objectives, test approach, and other relevant information. It helps in communicating the testing plan to stakeholders, including project managers, developers, and other team members. Clear documentation ensures that everyone involved in the testing process has a shared understanding of the goals, scope, and approach.

Overall, the goal of test planning is to ensure that testing activities are well-organized, efficient, and effective in achieving the desired testing objectives within the given constraints and project requirements. It provides a roadmap for the testing team, guides their efforts, and helps in delivering a high-quality software product or system.

## High Level Expectations

- Explicit expectations. ...
- Implicit expectations. ...
- Interpersonal expectations. ...
- Digital expectations. ...
- Dynamic performance expectations. ...
- Fast Customer Service. ...
- Accurate Data by Self-Service. ...
- Easy-to-Use Websites and Apps.

High-level expectations refer to overarching goals or outcomes that are set for a project, task, or individual. These expectations are typically broad and strategic, outlining the desired results rather than specific details on how to achieve them. Here are a few examples of high-level expectations:

**Performance:** A high-level expectation for an employee could be to consistently meet or exceed performance targets set by the company. This expectation focuses on the overall results and outcomes achieved by the employee.

**Quality:** For a product development project, a high-level expectation might be to deliver a high-quality product that meets customer requirements and industry standards. This expectation emphasizes the overall quality of the final deliverable.

**Customer Satisfaction:** In a customer service role, a high-level expectation may be to ensure a high level of customer satisfaction by providing timely and effective assistance. The emphasis here is on delivering exceptional customer service experiences.

**Innovation:** An expectation for a research and development team could be to foster a culture of innovation and consistently generate new ideas or solutions. This expectation encourages creativity and the exploration of new possibilities.

**Collaboration:** In a team setting, a high-level expectation might be to promote collaboration and effective communication among team members. This expectation emphasizes the importance of working together to achieve common goals.

**Growth and Development:** An expectation for individual employees could be to continuously learn and develop new skills to enhance their professional growth. This expectation encourages self-improvement and ongoing learning.

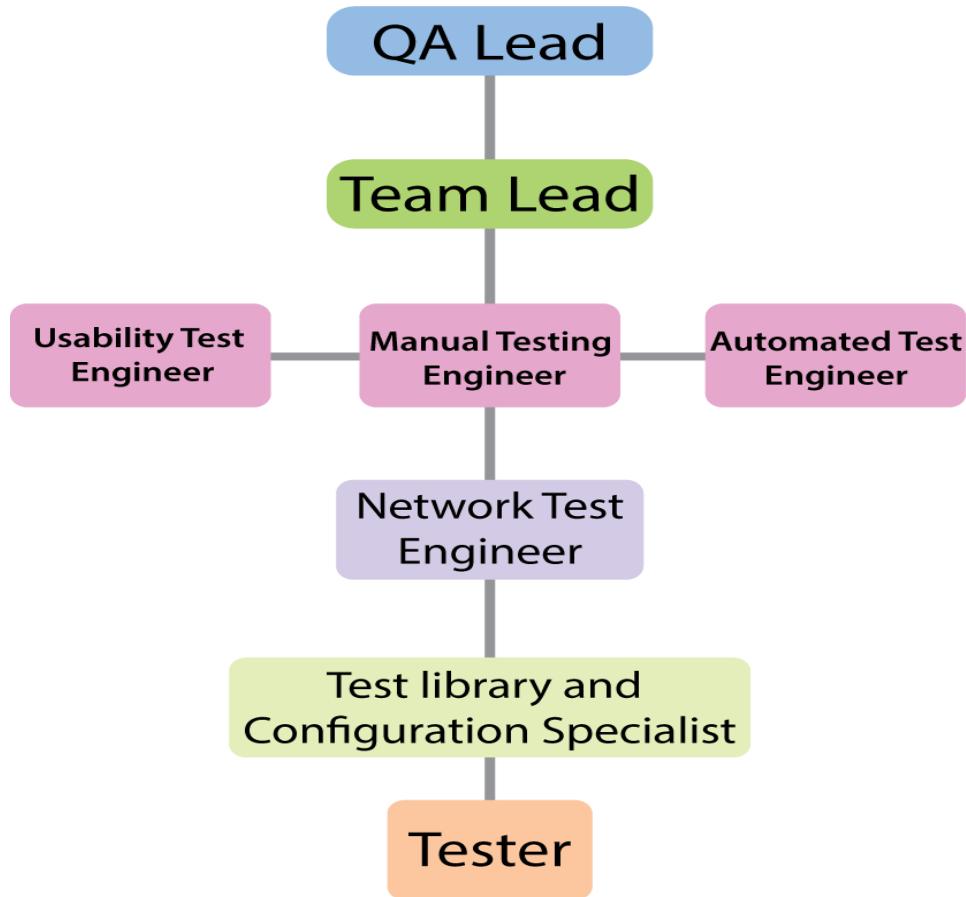
It's important to note that high-level expectations should be clear, measurable, and aligned with the overall goals and vision of the organization or project. They serve as guiding principles to help individuals and teams understand what is expected of them and to focus their efforts on achieving the desired outcomes.

## **Intergroup Responsibilities**

Software testing is an essential part of the software development life cycle (**SDLC**). Playing a significant role in defining the success rate of a particular product, owing to the same reason the software testing team plays a crucial role even after the product's development is completed

Therefore, it is important to ensure that this software testing team includes a perfect mix of talented as well as capable professionals who are also **domain experts**.

Being experts in the problem domain make it easier for them to create such test scripts that make it easier to identify the problem in the product.



While every company follows a different structure of the testing team, there are a few members who are common in every structure and fulfill the expectations of the team.

This includes:

#### 1. QA Leader:

QA Leader is the most important member of the testing team. While it is extremely crucial for him/her to have a clear understanding of the **testing process** or methodology. It is also essential for him/her to be familiar with the varied test-program concerns such as test environment and data management, trouble reporting and resolution, etc.

#### The Main Roles and Responsibilities handled by the QA leader are:

- Acts as a point of contact for inter and intra departmental interaction
- Represents the software testing team as well as enables customer relationship
- Deciding the test budget and schedule
- Identifying the testing activities for other team members like testers or test engineers
- Planning the entire testing process
- Checking the availability of the resources to execute testing activities
- Identifying if the process of testing is going in sync with the software development
- Preparing the status report of testing activities
- Sharing updates on testing with the project manager
- Planning pre and post-test meetings

## 2. Test Lead

With a clear understanding about the applications business area and its requirements, a test lead is a person who is also familiar with the varied test-program issues such as test data management, test design, and test development.

His/her expertise in numerous technical skills such as **programming languages**, database technologies, and computer operating systems also enable him/her to deliver the best at his/her job.

### **The Major Role and Responsibilities of a Test Lead include the following:**

- Technical expertise related to the test program and approach.
- Provides support for customer interface, staff planning, and supervision, as well as progress status reporting.
- Validating the quality of the testing requirements such as testability, test design, and script, **test automation**, etc.
- Staying updated about the latest test approaches and **tools**
- Assisting the software testing team to be aware of the latest trends in the world of software testing.
- Arranging walk-through for test design and procedure.
- Implementing the test process.
- Ensuring that test-product documentation is complete.

## 3. Test Engineer

The role of a test engineer is to determine the best way to create a process that can enable one to test a particular product in the best possible manner.

Test engineers can have different expertise based on which they are assigned a role in  
GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY    R KANNAN AP/CSE    74

a company.

Some of the common test engineers working in an organization are as mentioned below:

### **a) Usability Test Engineer**

These engineers are highly proficient in designing test suites as well as have a clear understanding of the usability issues. With excellent interpersonal skills, they are also skilled in test facilitation. Some of their common job roles include:

- Designing the **usability testing** scenarios
- Administering the process of usability testing
- Developing test-product documentation
- Participating in test-procedure walk-through

### **b) Manual Test Engineer**

With a clear understanding of the Graphical User Interface (GUI) design and its standards, manual test engineers are **highly proficient** in designing test suites and various testing techniques. Some of the major responsibilities of these engineers include:

- Using associated test data to design and develop test procedures and cases
- Manually executing the test procedures
- Attending test-procedure walk-through
- Following the required set standards

### **c) Automated Test Engineer**

Also known as Automater/developer, these engineers also have a good understanding of the GUI design and software testing. They can also be relied upon for designing the effective test suites as well as efficiently working with **test tools**. Some of the common roles handled by them are:

- Designing and developing test procedures on the basis of requirements
- Following rest-design standards
- Attending test procedure walk-throughs
- Executing the tests and preparing reports for the same.

#### 4. Network Test Engineer

With a high level of proficiency and expertise in a variety of technical skills such as programming languages, database technologies, and computer operating systems, network test engineers are good at product evaluation and integration skills.

##### **Their Major Roles at an Organization include:**

- Performing network, database, and middle-ware testing
- Developing load and stress test designs, cases and procedures
- Implementing the **performance monitoring tools** on an ongoing basis
- Conducting load and stress test procedures

#### 5. Test Library and Configuration Specialist:

This job role requires one to have a network, database, and system administration skills along with expertise in technical skills including **programming languages**, database technologies, and computer operating systems. Their major job roles include the following:

- Managing the test-script change
- Maintaining test-script version control
- Upholding test-script reuse library
- Creating test builds, wherever required

#### 6. Tester

Having a sound knowledge about various concepts involved in test designing and execution methodologies, a software tester is the one who is able to interact efficiently with the development team. His/her major roles as a part of software testing team includes:

- **Designing** the testing scenarios for usability testing
- Analyzing the testing results and submitting the report to the development team
- Creating test designs, processes, cases and test-product documentation
- Conducting testing as per the set standards and procedures
- Ensure that the testing is carried out as per the defined standards and procedures

## Test Phases

Each phase of the software testing life cycle allows developers to assess specific characteristics of the software and evaluate whether the software is suitable for use. The various evaluations during testing can identify errors and deficits in the application before it enters production and deployment. Finding and resolving such issues early on helps preserve your reputation and clients' confidence in your work.

Early and effective software testing can also be financially beneficial. By allowing developers to address flaws in software design, functionality and security as soon as testers discover them, software testing spares the need for costly changes to the software while it's in wide use. Resolving such problems during development also helps ensure that customers have high regard for the software, potentially leading to increased sales.

## What are the 5 phases of testing software?

In the software testing life cycle, there are usually five phases of testing:

### 1. Static testing

During static testing, developers work to avoid potential problems that might arise later. Without executing the code, they perform manual or automated reviews of the supporting documents for the software, such as requirement specifications, searching for any potential ambiguities, errors or redundancies. The goal is to preempt defects before introducing them to the software system.

### 2. Unit testing

The next phase of software testing is unit testing. During this phase, the software undergoes assessments of its specific units, or its functions and procedures, to ensure that each works properly on its own. The developers may use white box testing to evaluate the software's code and internal structure, commonly before delivering the software for formal testing by testers. Unit testing can occur whenever a piece of code undergoes change, which allows for quick resolution of issues.

### 3. Integration testing

Integration testing involves testing all the units of a program as a group to find issues with how the separate software functions interact with one another. Through integration testing, the developers can determine the overall efficiency of the units as they run together. This phase is important because the program's overall functionality relies on the units operating simultaneously as a complete system, not as isolated procedures.

## 4. System testing

In the system testing phase, the software undergoes its first test as a complete, integrated application to determine how well it carries out its purpose. For this, the developers pass the software to independent testers who had no involvement in its development to ensure that the testing results stem from impartial evaluations. System testing is vital because it ensures that the software meets the requirements as determined by the client.

## 5. Acceptance testing

Acceptance testing is the last phase of software testing. Its purpose is to evaluate the software's readiness for release and practical use. Testers may perform acceptance testing alongside individuals who represent the software's target audience. Acceptance testing aims to show whether the software meets the needs of its intended users and that any changes the software experiences during development are appropriate for use. The representative individuals are crucial to this phase because they can offer insight into what customers may want from the software. Once the software passes acceptance testing, it moves on to production.

# 5 types of additional software tests

In addition to the five main phases of testing, you may also need additional tests to evaluate software. The necessity of these tests depends on the type of company you work for and the software you're creating. These tests include:

## 1. Performance testing

In performance testing, testers evaluate how well the software handles various scenarios and workloads. There are several subtypes of performance testing. A common performance test is load testing, which recreates real-life user conditions to determine how the software performs in common scenarios. Another test is stress testing, in which testers intentionally overload the software to discover how much it can handle before it fails.

## 2. Regression testing

Regression testing is a procedure that occurs throughout the testing life cycle. After developers implement a change to the software, testers perform regression testing to ensure that previously tested and functional operations remain intact. This is a good way to help guarantee consistency in functionality.

### 3. Usability testing

Usability testing focuses on ease of use. Testers approach the software from the perspective of end users, validating that its interface and design are simple to understand and that the application is easy to operate. Ideally, the user can learn the software on their own and enjoy a satisfying experience with it.

### 4. Compatibility testing

Compatibility testing evaluates the software's ability to function as intended in various computing environments. These include operating systems, mobile platforms and web browsers. Such environments have their own specifications for the software they run, so it's important to confirm that the software meets all of those differing specifications.

### 5. Security testing

Security testing is an assessment of the software in terms of threats, risks and vulnerabilities. Testers might examine the software for flaws that expose a user's personal data to hackers or make the software susceptible to malware. If testers find any such flaws, the developers can secure them with coding.

## Test Strategy

A high-level document is used to validate the test types or levels to be executed for the product and specify the **Software Development Life Cycle's** testing approach is known as Test strategy document.

Once the test strategy has been written, we cannot modify it, and it is approved by the **Project Manager, development team**.

The test strategy also specifies the following details, which are necessary while we write the test document:

- **What is the other procedure having to be used?**
- **Which module is going to be tested?**
- **Which entry and exit criteria apply?**
- **Which type of testing needs to be implemented?**

In other words, we can say that it is a document, which expresses how we go about testing the product. And the approaches can be created with the help of following aspects:

- **Automation or not**
- **Resource point of view**

We can write the test strategy based on **development design documents**.

The development design document includes the following documents:

- **System design documents:** Primarily, we will use these documents to write the test strategy.
- **Design documents:** These documents are used to specify the software's functionality to be enabled in the upcoming release.
- **Conceptual design documents:** These are the document which we used Infrequently.

### The Objective of Test Strategy

- The primary objective of writing the test strategy is to make sure that all purposes are covered entirely and understood by all stakeholders, we should systematically create a test strategy.
- Furthermore, a test strategy objective is to support various quality assurance stockholders in respect of **planning of resources, language, test and integration levels, traceability, roles and responsibilities**, etc

### Features of Test Strategy Document

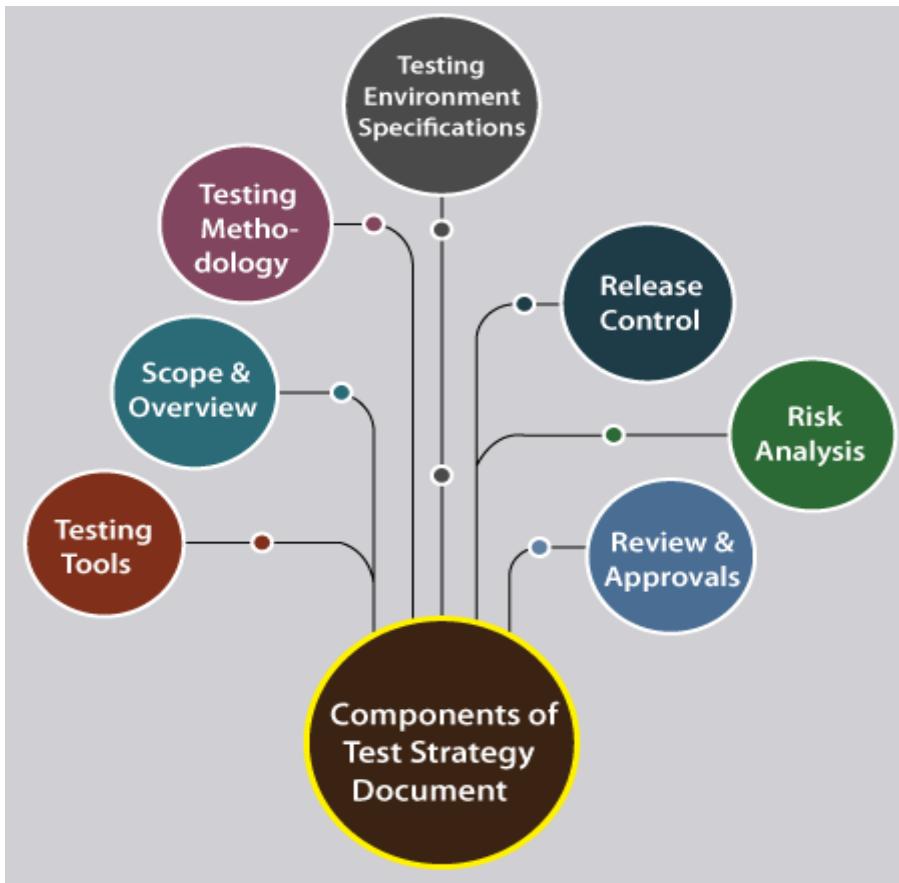
- In **SDLC (Software Development Life Cycle)**, the test strategy document plays an important role. It includes various significant aspects, such as who will implement the testing, what will be tested, how it will be succeeded, and what risks and incidents will be related to it.
- Some of the additional characteristics of the Test Strategy document are as follows:
  - **Test Team Lead**
  - **Development Manager**
  - **Quality Analyst Manager**
  - **Product Manager**
- The test strategy document is approved and reviewed by the following's peoples:
  - **Test Team Lead**
  - **Development Manager**
  - **Quality Analyst Manager**
  - **Product Manager**

- For different testing activities, the test strategy document specifies the resources, scope, plan, methodology, etc.
- In order to direct how testing will be achieved, it is used by the project test team once it is ready or completed.
- Primarily, it is obtained from the **BRS (Business Requirements Specifications)** documents.
- The test strategy document is a high-level document, which generally remains constant, implying no frequent and pointless modification is made in the document.
- The respective team easily accomplishes the objectives of testing with the help of a test strategy document.
- The respective team easily accomplishes the objectives of testing with the help of test strategy document.

### Components of Test Strategy Document

We understand that the test strategy document is made during the requirements phase and after the requirements have been listed.

Like other testing documents, the test strategy document also includes various components, such as:



- **Scope and Overview**
- **Testing Methodology**
- **Testing Environment Specifications**
- **Testing Tools**
- **Release Control**
- **Risk Analysis**
- **Review and Approvals**

Let's see them one by one for our better understanding:

### 1. Scope and Overview

- The first component of the test strategy document is **Scope and Overview**.
- The overview of any product contains the information on who should approve, review and use the document.

- The test strategy document also specified the testing activities and phases that are needed to be approved.

## 2. Testing Methodology

- The next module in the test strategy document is **Testing methodology**, which is mainly used to specify **the levels of testing, testing procedure, roles, and responsibilities** of all the team members.
- The testing approach also contains the change management process involving the modification request submission, pattern to be used, and activity to manage the request.
- Above all, if the test strategy document is not established appropriately, then it might lead to **errors or mistakes** in the future.

## 3. Testing Environment Specifications

- Another component of the test strategy document is **Testing Environment Specification**.
- As we already aware of the specification of the **test data requirements** is exceptionally significant. Hence, clear guidelines on how to prepare test data are involved in the testing environment specification of the test strategy document.
- This module specifies the information related to **the number of environments and the setup demanded**.
- The backup and restore strategies are also offered to ensure that there is no data loss because of the coding or programming issues.

## 4. Testing Tools

- **Testing tools** are another vital component of the test strategy document, as it stipulates the complete information about the **test management and automation tools** necessary for test execution activity.;
- For **security, performance, load testing**, the necessary methodologies, and tools are defined by the details of the **open-source or commercial tool** and the number of users that can be kept by it.

## 5. Release Control

- Another important module of the test strategy document is **Release Control**.

- It is used to ensure that the correct and effective **test execution** and release management strategies should be systematically developed.

## 6. Risk Analysis

- The next component of the test strategy document is **Risk Analysis**.
- In the test strategy document, all the possible risks are described linked to the project, which can become a problem in test execution.
- Furthermore, for inclining these risks, a clear strategy is also formed in order to make sure that they are undertaking properly.
- We also create a contingency plan if the development team faces these risks in real-time.

## 7. Review and Approvals

- The last component of the Testing strategy document is **Review and Approval**.
- When all the related testing activities are specified in the test strategy document, it is reviewed by the concerned people like:
  - **System Administration Team**
  - **Project Management Team**
  - **Development Team**
  - **Business Team**
- Together with the **correct date, approver name, comment, and summary** of the reviewed variations should be followed while starting the document.
- Likewise, it should be constantly reviewed and updated with the testing process improvements.

## Types of Test Strategies

Here, we are discussing some of the significant types of test strategies document:

## Types of Test Strategies



- **Methodical strategy**
- **Reactive strategy**
- **Analytical strategy**
- **Standards compliant or Process compliant strategy**
- **Model-based strategy**
- **Regression averse strategy**
- **Consultative strategy**

Let's understand them one by one in detail:

### 1. Methodical Strategy

- The first part of test strategy document is **Methodical strategy**.
- In this, the test teams follow a **set of test conditions, pre-defined quality standard**(like ISO25000), **checklists**.
- The Standard checklists is occurred for precise types of testing, such as **security testing**.

### 2. Reactive Strategy

- The next type of test strategy is known as **Reactive strategy**.
- In this, we can design the test and execute them only after the real software is delivered, Therefore, the **testing is based upon the identified defects**in the existing system.

- Suppose, we have used the **exploratory testing**, and the test approvals are established derived from the existing aspects and performances.
- These test approvals are restructured based on the outcome of the testing which is implemented by the test engineer.

### 3. Analytical strategy

- Another type of test strategy is **Analytical strategy**, which is used to perform testing based on requirements, and requirements are analyzed to derive the test conditions. And then **tests are designed, implemented, and performed** to encounter those requirements. For example, **risk-based testing or requirements-based testing**.
- Even the outcomes are recorded in terms of **requirements**, such as **requirements tested and passed**.

### 4. Standards compliant or Process compliant strategy

- In this type of test strategy, the test engineer will follow the **procedures or guidelines created by a panel of industry specialists** or **committee standards** to find test conditions, describe test cases, and put the testing team in place.
- Suppose any project follows the **ScrumAgile** technique. In that case, the test engineer will generate its complete test strategy, beginning from classifying test criteria, essential test cases, performing tests, report status, etc., around each **user story**.
- Some **good examples** of the standards-compliant process are **Medical systems following US FDA (Food and Drugs Administration) standards**.

### 5. Model-based strategy

- The next type of test strategy is a **model-based strategy**. The testing team selects the **current or expected situation** and produces a model for it with the following aspects: inputs, **outputs, processes, and possible behavior**.
- And the models are also established based on the current data speeds, software, hardware, infrastructure, etc.

## 6. Regression averse strategy

- In the regression averse strategy, the test engineer mainly **emphasizes decreasing regression risks for functional or non-functional product shares.**
- **For example,** suppose we have one web application to test the regression issues for the particular application. The testing team can develop the test automation for both **typical and exceptional use cases** for this scenario.
- And to facilitate the tests can be run whenever the application is reformed, the testing team can use **GUI-based automation tools.**

## 7. Consultative strategy

- The consultative strategy is used to **consult key investors as input to choose the scope** of test conditions as in user-directed testing.
- In order of priority, the client will provide a list of **browsers and their versions, operating systems, a list of connection types, anti-malware software**, and also the contradictory list, which they want to test the application.
- As per the need of the items given in provided lists, the test engineer may use the various testing techniques, such as **equivalence partitioning**

We can combine the two or more strategies as per the needs of the product and organization's requirements. And it is not necessary to use any one of the above listed test strategies for any testing project.

### Test strategy selection

The selection of the **test strategy** may depend on the below aspects:

- The selection of test strategy depends on the **Organization type and size.**
- We can select the test strategy based on the **Project requirements**, such as **safety and security** related applications require rigorous strategy.
- We can select the test strategy based on the **Product development model.**

The final document of the test strategy contains important details about the following factors:

- Scope and Overview
- Re-usability of both software and testing work products.

- Details of different Test levels, relationships between the test levels, and procedure to integrate different test levels.
- Testing environment
- Testing techniques
- Level of automation for testing
- Different testing tools
- Risk Analysis
- For each test level Entry as well exit conditions
- Test results reports
- Degree of independence of each test
- Metrics and measurements to be evaluated during testing
- Confirmation and regression testing
- Managing defects detected
- Managing test tools and infrastructure configuration
- Roles and responsibilities of Test team members

### Conclusion

After understanding the **test strategy document**, at last, we can say that the test strategy document provides a vibrant vision of what the test team will do for the whole project.

The test strategy document could prepare only those who have good experience in the **product domain** because the test strategy document will drive the entire team.

And it cannot be modified or changed in the complete project life cycle as it is a static document.

Before any testing activities begin, the Test strategy document can distribute to the entire testing team.

If the test strategy document is written correctly, it will develop a high-quality system and expand the complete testing process.

### Resource Requirements

Resource requirement is a detailed summary of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project.

The resource requirement and planning is important factor of the test planning because helps in determining the number of resources (employee, equipment...) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

Some of the following factors need to be considered:

- Machine configuration (RAM, processor, disk) needed to run the product under test.
- Overheads required by test automation tools, if any
- Supporting tools such as compilers, test data generators, configuration management tools.
- The different configurations of the supporting software (e.g. OS) that must be present
- Special requirements for running machine-intensive tests such as load tests and performance tests.
- Appropriate number of licenses of all the software

| No. | Member            | Tasks                                                                                                                                                                                                                                                                                                                                                            |
|-----|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | Test Manager      | Manage the whole project Define project directions Acquire appropriate resources                                                                                                                                                                                                                                                                                 |
| 2   | Tester            | Identifying and describing appropriate test techniques/tools/automation architecture Verify and assess the Test Approach Execute the tests, Log results, Report the defects. Tester could be in-sourced or out-sourced members, base on the project budget For the task which required low skill, I recommend you choose outsourced members to save project cost |
| 3.  | Developer in Test | Implement the test cases, test program, test suite etc.                                                                                                                                                                                                                                                                                                          |

| No. | Member             | Tasks                                                                                                                                      |
|-----|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 4.  | Test Administrator | Builds up and ensures test environment and assets are managed and maintained Support Tester to use the test environment for test execution |
| 5.  | SQA members        | Take in charge of quality assurance Check to confirm whether the testing process is meeting specified requirements                         |

**OR**

**Human Resource:** The following table represents various members in your project team

**System Resource:** For testing, a web application, you should plan the resources as following tables:

| No. | Resources | Descriptions                                                                                                                                                                               |
|-----|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | Server    | Install the web application under test This includes a separate web server, database server, and application server if applicable                                                          |
| 2   | Test tool | The testing tool is to automate the testing, simulate the user operation, generate the test results There are tons of test tools you can use for this project such as Selenium, QTP...etc. |
| 3.  | Network   | You need a Network include LAN and Internet to simulate the real business and user environment                                                                                             |

| No. | Resources | Descriptions                                           |
|-----|-----------|--------------------------------------------------------|
| 4.  | Computer  | The PC which users often use to connect the web server |

## Tester Assignments

They may be involved in or even be the primary people identifying test conditions and creating test designs, test cases, test procedure specifications and test data, and may automate or help to automate the tests.

Tester assignments typically refer to the process of assigning individuals or teams to test specific components, features, or aspects of a product or system. Testing is an essential part of software development, quality assurance, and product validation. Assigning testers to different tasks helps ensure thorough and efficient testing coverage. Here are some considerations for tester assignments:

**Test plan and strategy:** Begin by creating a test plan and strategy that outlines the testing objectives, scope, and approach. This will help identify the different types of tests required and the specific areas that need to be covered.

**Test case creation:** Testers can be assigned to create test cases based on the test plan and requirements. Test cases should cover different scenarios and use cases to ensure comprehensive testing.

**Testing types:** Identify the different types of testing needed, such as functional testing, performance testing, security testing, usability testing, etc. Assign testers with the relevant expertise and skills for each type of testing.

**Test environment:** Assign testers to set up and configure the test environment, including any necessary hardware, software, or network configurations. This ensures that the testing environment accurately reflects the production environment.

**Test execution:** Assign testers to execute the test cases and document the results. Testers should follow the test plan and report any issues or bugs they encounter during the testing process.

**Bug reporting and tracking:** Assign testers to report identified bugs or issues in a structured manner, including detailed descriptions, steps to reproduce, and any supporting documentation. Testers may also be responsible for tracking the status and resolution of reported issues.

**Regression testing:** After bug fixes or changes are made, assign testers to perform regression testing to ensure that the fixes or changes did not introduce new issues or break existing functionality.

**Collaboration and communication:** Assign testers to collaborate with developers, product managers, and other stakeholders to ensure clear communication and understanding of testing requirements, priorities, and progress.

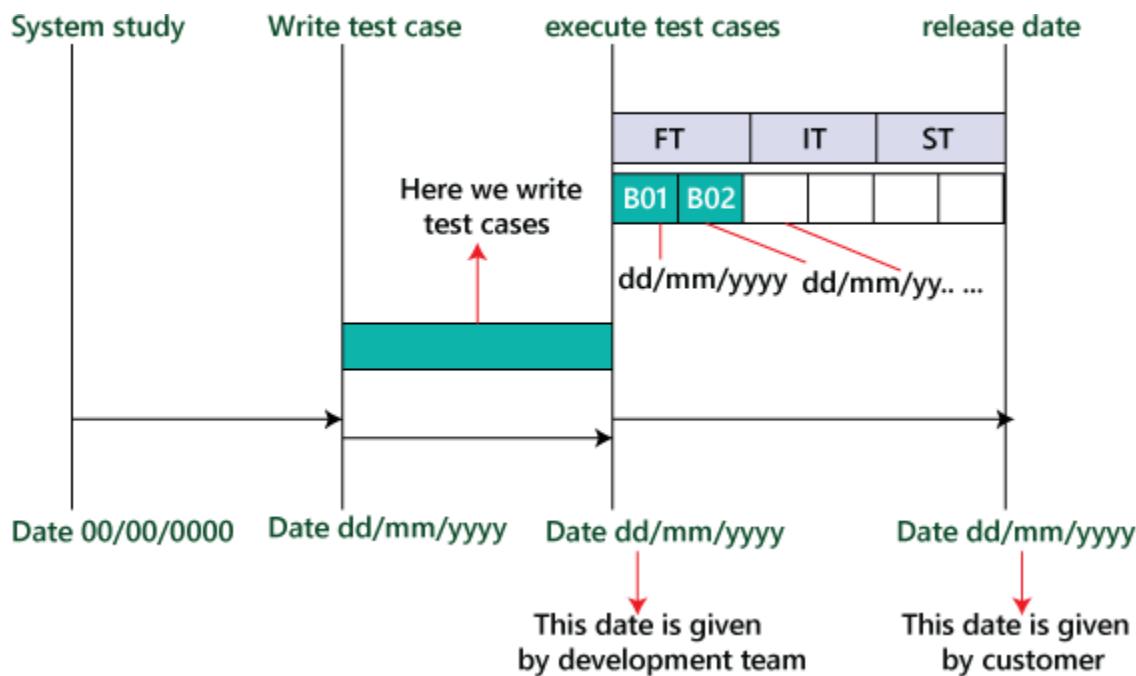
**Test coverage analysis:** Assign testers to analyze the test coverage and identify any gaps or areas that require additional testing. This helps ensure that all critical features and functionalities are adequately tested.

**Test automation:** Assign testers with automation skills to develop and maintain automated test scripts, which can help increase testing efficiency and coverage.

It's important to consider the skills, experience, and availability of testers when making assignments. Regular communication and coordination among testers and other team members are crucial to ensure effective testing and timely feedback.

## Test Schedule

It is used to explain the timing to work, which needs to be done or this attribute covers when exactly each testing activity should start and end? And the exact date is also mentioned for every testing activity for the particular date.



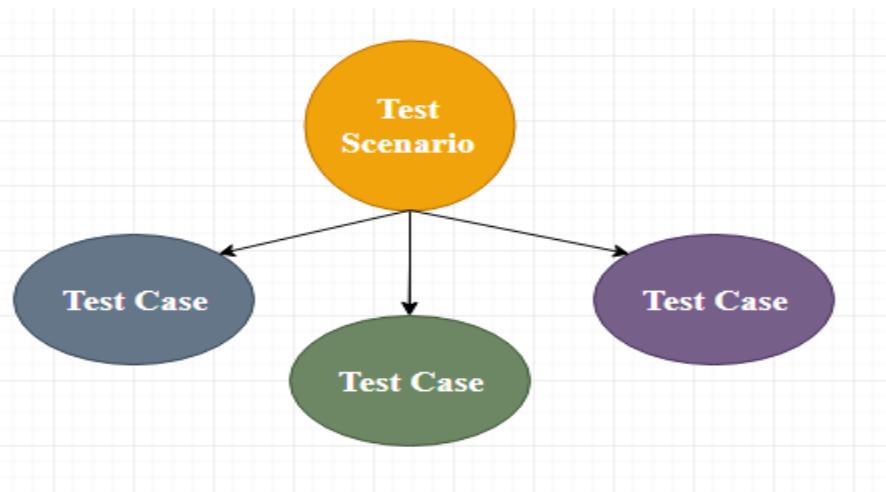
Therefore as we can see in the below image that for the particular activity, there will be a starting date and ending date; for each testing to a specific build, there will be the specified date.

**For example**

- Writing test cases
- Execution process

**Test Cases**

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-detailed document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process. Writing of test cases is a one-time attempt that can be used in the future at the time of regression testing.

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.

### 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 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.
- The test engineers never look at the requirement while testing the product document because testing is constant and does not depends on the mood of the person rather than the quality of the test engineer.

### Why we write the test cases?

We will write the test for the following reasons:

- **To require consistency in the test case execution**
- **To make sure a better test coverage**
- **It depends on the process rather than on a person**
- **To avoid training for every new test engineer on the product**

**To require consistency in the test case execution:** we will see the test case and start testing the application.

**To make sure a better test coverage:** for this, we should cover all possible scenarios and document it, so that we need not remember all the scenarios again and again.

**It depends on the process rather than on a person:** A test engineer has tested an application during the first release, second release, and left the company at the time of third release. As the test engineer understood a module and tested the application thoroughly by deriving many values. If the person is not there for the third release, it becomes difficult for the new person. Hence all the derived values are documented so that it can be used in the future.

**To avoid giving training for every new test engineer on the product:** When the test engineer leaves, he/she leaves with a lot of knowledge and scenarios. Those scenarios should be documented so that the new test engineer can test with the given scenarios and also can write the new scenarios.

### Test case template

The primary purpose of writing a test case is to achieve the efficiency of the application.

#### Header

Test Case Name/ID :- **Release - Version - Application Name - Module**

Test Case Type:- **F.T.C    I.T.C    S.T.C**

Requirement Number:-

Module:-

Severity:- **Critical/Major/Minor**

Status:-

Release:-

Version:-

Pre-condition:-

Test Data:-

Summary:-

#### Body

| Step No. | Description | Inputs | Expected Result | Actual Result | Status | Comments |
|----------|-------------|--------|-----------------|---------------|--------|----------|
| ...      | ...         | ...    | ...             | ...           | ...    | ...      |
| ...      | ...         | ...    | ...             | ...           | ...    | ...      |

#### Footer

{ Author:-              Reviewd By:-

                Date:-              Approved By:-

As we know, the **actual result** is written after the test case execution, and most of the time, it would be same as the **expected result**. But if the test step will fail, it will be different. So, the actual result field can be skipped, and in the **Comments** section, we can write about the bugs.

And also, the **Input field** can be removed, and this information can be added to the **Description field**.

The above template we discuss above is not the standard one because it can be different for each company and also with each application, which is based on the test engineer and the test lead. But, for testing one application, all the test engineers should follow a usual template, which is formulated.

The test case should be written in simple language so that a new test engineer can also understand and execute the same.

In the above sample template, the header contains the following:

### **Step number**

It is also essential because if step number 20 is failing, we can document the bug report and hence prioritize working and also decide if it's a critical bug.

### **Test case type**

It can be functional, integration or system test cases or positive or negative or positive and negative test cases.

### **Release**

One release can contain many versions of the release.

### **Pre-condition**

These are the necessary conditions that need to be satisfied by every test engineer before starting the test execution process. Or it is the data configuration or the data setup that needs to be created for the testing.

**For example:** In an application, we are writing test cases to add users, edit users, and delete users. The per-condition will be seen if user A is added before editing it and removing it.

### **Test data**

These are the values or the input we need to create as per the per-condition.

**For example,** Username, Password, and account number of the users.

The test lead may be given the test data like username or password to test the application, or the test engineer may themselves generate the username and password.

### Severity

The severity can be **major, minor, and critical**, the severity in the test case talks about the importance of that particular test cases. All the test execution process always depends on the severity of the test cases.

We can choose the severity based on the module. There are many features included in a module, even if one element is critical, we claim that test case to be critical. It depends on the functions for which we are writing the test case.

**For example,** we will take the Gmail application and let us see the severity based on the modules:

| Modules      | Severity |
|--------------|----------|
| Login        | Critical |
| Help         | Minor    |
| Compose mail | Critical |
| Setting      | Minor    |
| Inbox        | Critical |
| Sent items   | Major    |
| Logout       | Critical |

And for the banking application, the severity could be as follows:

| Modules         | Severity |
|-----------------|----------|
| Amount transfer | critical |
| Feedback        | minor    |

### Brief description

The test engineer has written a test case for a particular feature. If he/she comes and reads the test cases for the moment, he/she will not know for what feature has written it. So, the brief description will help them in which feature test case is written.

### Example of a test case template

Here, we are writing a test case for the **ICICI application's Login module**:

| New Microsoft Excel Worksheet - Microsoft Excel |                                           |                                            |                   |                                        |                               |             |
|-------------------------------------------------|-------------------------------------------|--------------------------------------------|-------------------|----------------------------------------|-------------------------------|-------------|
| File                                            |                                           | Home                                       | Insert            | Page Layout                            | Formulas                      | Data        |
| Cut                                             | Copy                                      | Format Painter                             | Font              | Wrap Text                              | General                       | Conditional |
| Paste                                           | Font                                      | Font                                       | Font              | Font                                   | Font                          | Font        |
| Clipboard                                       | Font                                      | Font                                       | Font              | Font                                   | Font                          | Font        |
| DS                                              | fx                                        | Font                                       | Font              | Font                                   | Font                          | Font        |
| A                                               | B                                         | C                                          | D                 | E                                      | F                             | G           |
| 1                                               | Test case template                        |                                            |                   |                                        |                               |             |
| 2                                               | test case name                            | Delta-3.0-ICICI-Login                      |                   |                                        |                               |             |
| 3                                               | test case type                            | Functional test case                       |                   |                                        |                               |             |
| 4                                               | requirement no                            |                                            | 1                 |                                        |                               |             |
| 5                                               | module                                    | login                                      |                   |                                        |                               |             |
| 6                                               | status                                    | XXX                                        |                   |                                        |                               |             |
| 7                                               | severity                                  | critical                                   |                   |                                        |                               |             |
| 8                                               | release                                   | Delta                                      |                   |                                        |                               |             |
| 9                                               | version                                   |                                            | 3                 |                                        |                               |             |
| 10                                              | pre-condition                             | required one login                         |                   |                                        |                               |             |
| 11                                              | test data                                 | username-abc, password-123                 |                   |                                        |                               |             |
| 12                                              | summary                                   | to check the functionality of login        |                   |                                        |                               |             |
| 13                                              | Steps no                                  | Description                                | Inputs            | Expected result                        | Actual results                | Status      |
| 14                                              | 1                                         | open "Browser" and enter the "Url"         | https://QA/Main// | Login page must be display             | As Expected                   | pass        |
| 15                                              | 2                                         | enter the following values for "Username": |                   |                                        |                               | XXX         |
| 16                                              | Valid(abc)                                | abc                                        | Accpect           |                                        | Login page must be disp pass  | XXX         |
| 17                                              | Invalid                                   |                                            | 555               | Error message "invalid login"          | not as expected               | fail        |
| 18                                              | Blank                                     | Null                                       |                   | Error message username cannot be empty | not as expected               | fail        |
| 19                                              | Symbols                                   | 2 alphabet                                 |                   | Error message invalid login            | not as expected               | fail        |
| 20                                              | 3                                         | enter the following values for "Password": |                   |                                        |                               |             |
| 21                                              | valid                                     |                                            | 123               | Accpect                                | Login page must be disp pass  | XXX         |
| 22                                              | invalid                                   | xy3                                        |                   | Error message invalid login            | not as expected               | fail        |
| 23                                              | Blank                                     |                                            |                   | Error message password cannot be empty | not as expected               | fail        |
| 24                                              | enter the valid username and password and |                                            |                   |                                        |                               |             |
| 25                                              | 4 click on "OK" button                    | abc,123                                    |                   | "home Pag " must be displayed          | home page is displayin pass   |             |
| 26                                              | enter the valid username and password and |                                            |                   |                                        |                               |             |
| 27                                              | 5 click on "Cancel" button                | abc,123                                    |                   | all field must be cleared              | the entered data is clea pass | XXX         |
| 28                                              | author                                    | test engineer name                         |                   |                                        |                               |             |
| 29                                              | date                                      |                                            | 1/4/2020          |                                        |                               |             |
| 30                                              | reviewed by                               | ryan                                       |                   |                                        |                               |             |
| 31                                              | apporved by                               | jessica                                    |                   |                                        |                               |             |

## Types of test cases

We have a different kind of test cases, which are as follows:

- **Function test cases**
- **Integration test cases**
- **System test cases**

### The functional test cases

Firstly, we check for which field we will write test cases and then describe accordingly.

In functional testing or if the application is data-driven, we require the input column else; it is a bit time-consuming.

**Rules to write functional test cases:**

- In the expected results column, try to use **should be** or **must be**.
- Highlight the Object names.
- We have to describe only those steps which we required the most; otherwise, we do not need to define all the steps.
- To reduce the excess execution time, we will write steps correctly.
- Write a generic test case; do not try to hard code it.

Let say it is the amount transfer module, so we are writing the functional test cases for it and then also specifies that it is not a login feature.

The form is titled "AMOUNT TRANSFER". It contains three input fields: "From Account Number" (labeled "(FAN)", with five placeholder dots on the left), "To Account Number" (labeled "(TAN)", with five placeholder dots on the left), and "Amount" (with no placeholder text). At the bottom are two buttons: "TRANSFER" and "CANCEL".

The functional test case for amount transfer module is in the below Excel file:

| Functional Test case template |                                                                                       |                    |                                                                       |                |        |          |
|-------------------------------|---------------------------------------------------------------------------------------|--------------------|-----------------------------------------------------------------------|----------------|--------|----------|
| Steps no                      | Description                                                                           | Inputs             | Expected result                                                       | Actual results | Status | Comments |
| 1                             | Open "Browser" and enter the "Url"                                                    | https://QA/Main/IV | "Login page" must be display                                          | As Expected    | pass   | XXX      |
| 2                             | Enter the following values for "Username" and "Password" and click on the "OK" button | xyz, 1234          | "Home page" must be displayed                                         | As Expected    | pass   | XXX      |
| 3                             | Click on the "Amount Transfer"                                                        | Null               |                                                                       |                | pass   | XXX      |
| 4                             | Enter the following for From Account number(FAN):                                     |                    |                                                                       |                |        |          |
|                               | valid                                                                                 | 1234               | Accept                                                                | As Expected    | pass   |          |
|                               | invalid                                                                               | 1124               | Error message invalid account                                         |                | fail   |          |
|                               | blank                                                                                 | —                  | Error message FAN value cannot be blank                               |                | fail   |          |
|                               | —                                                                                     | —                  | test maximum coverage                                                 |                |        |          |
| 5                             | Enter the following values for "TO account number"(TAN)                               |                    |                                                                       |                |        |          |
|                               | valid                                                                                 | 4321               | Accept                                                                | As expected    | pass   | XXX      |
|                               | invalid                                                                               | 6665               | Error message invalid account                                         |                | fail   |          |
|                               | Blank                                                                                 | —                  | Error message TAN value cannot be blank                               |                |        |          |
|                               | —                                                                                     | —                  | test maximum coverage                                                 |                |        |          |
| 6                             | enter the value for "Amount"                                                          |                    |                                                                       |                |        |          |
|                               | valid                                                                                 | 5000, 5001,9000,84 | Accept                                                                | As expected    | pass   | XXX      |
|                               | invalid                                                                               | 4999,,9001         | error message amount should be between (5000-9000)                    |                | fail   |          |
| 7                             | Enter the value for "FAN, TAN, Amount"click on the "Transfer" button                  |                    |                                                                       |                |        |          |
|                               | FAN                                                                                   | 1234               |                                                                       |                |        |          |
|                               | TAN                                                                                   | 4321               | "Confirmation Message" amount transfer successfully must be displayed | As expected    | pass   | XXX      |
|                               | Amount                                                                                | 6000               |                                                                       |                |        |          |
| 8                             | Enter the value for "FAN, TAN, Amount"click on the "Cancel" button                    |                    |                                                                       |                |        |          |
|                               | FAN                                                                                   | 1234               |                                                                       |                |        |          |
|                               | TAN                                                                                   | 4321               | All field must be cleared                                             | As expected    | pass   | XXX      |
|                               | Amount                                                                                | 6000               |                                                                       |                |        |          |
| Author                        | Sern                                                                                  |                    |                                                                       |                |        |          |
| Date                          |                                                                                       | 4/1/2020           |                                                                       |                |        |          |
| Reviewed by                   | jessica                                                                               |                    |                                                                       |                |        |          |
| Approved by                   | ryan                                                                                  |                    |                                                                       |                |        |          |

## Integration test case

In this, we should not write something which we already covered in the functional test cases, and something we have written in the integration test case should not be written in the system test case again.

## Rules to write integration test cases

- Firstly, understand the product
- Identify the possible scenarios
- Write the test case based on the priority

When the test engineer writing the test cases, they may need to consider the following aspects:

If the test cases are in details:

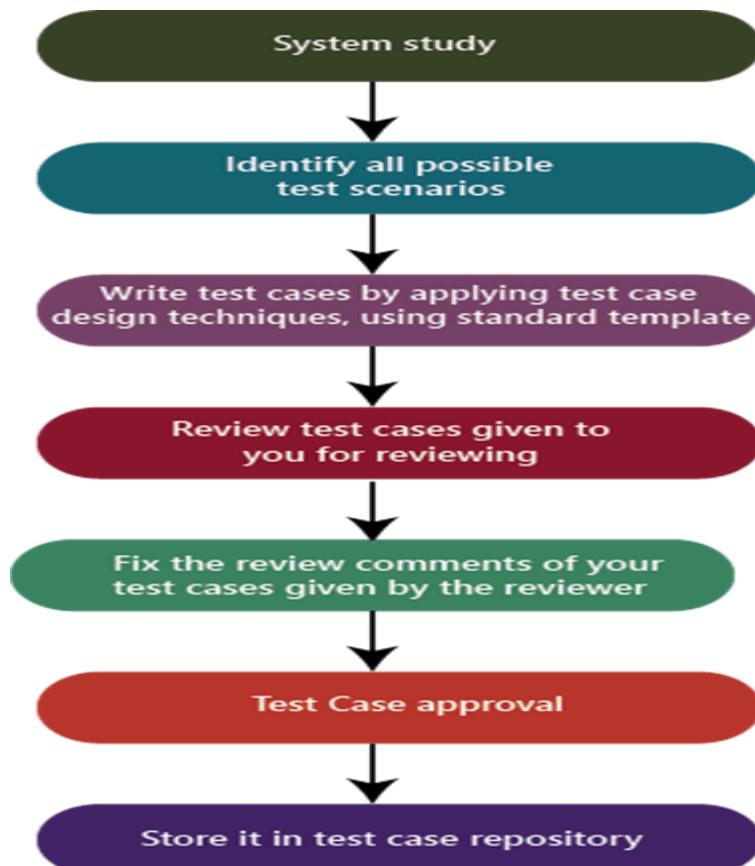
- They will try to achieve maximum test coverage.
- All test case values or scenarios are correctly described.
- They will try to think about the execution point of view.
- The template which is used to write the test case must be unique.

### System test cases

We will write the system test cases for the end-to-end business flows. And we have the entire modules ready to write the system test cases.

### The process to write test cases

The method of writing a test case can be completed into the following steps, which are as below:



## System study

In this, we will understand the application by looking at the requirements or the SRS, which is given by the customer.

### Identify all scenarios:

- When the product is launched, what are the possible ways the end-user may use the software to identify all the possible ways.
- I have documented all possible scenarios in a document, which is called test design/high-level design.
- The test design is a record having all the possible scenarios.

### Write test cases

Convert all the identified scenarios to test claims and group the scenarios related to their features, prioritize the module, and write test cases by applying test case design techniques and use the standard test case template, which means that the one which is decided for the project.

### Review the test cases

Review the test case by giving it to the head of the team and, after that, fix the review feedback given by the reviewer.

### Test case approval

After fixing the test case based on the feedback, send it again for the approval.

### Store in the test case repository

After the approval of the particular test case, store in the familiar place that is known as the test case repository.

## **Metrics and Statistics.**

Software Testing metrics are quantitative steps taken to evaluate the software testing process's quality, performance, and progress. This helps us to accumulate reliable data about the software testing process and enhance its efficiency. This will allow developers to make proactive and precise decisions for upcoming testing procedures.

What is a metric in software testing metrics?

A Metric is a degree to which a system or its components retains a given attribute. Testers don't define a metric just for the sake of documentation. It serves greater purposes in software testing. For example, developers can apply a metric to assume the time it takes to develop software. It can also be assigned to determine the numbers of new features and modifications, etc., added to the software.

**Importance of Software Testing Metrics**

As mentioned, test metrics are crucial to measuring the quality and performance of the software. With proper software testing metrics, developers can–

- Determine what types of improvements are required to deliver a defect-free, quality software
- Make sound decisions about the subsequent testing phases, such as scheduling upcoming projects as well as estimating the overall cost of those projects
- Evaluate the current technology or process and check whether it needs further modifications

**Types of software testing metrics**

There are three types of software testing metrics–

- Process Metrics:** Process metrics define the characteristics and execution of a project. These characteristics are essential to the improvement and maintenance of the process in the SDLC (Software Development Life Cycle).
- Product Metrics:** Product metrics define the size, design, performance, quality, and complexity of a product. By using these characteristics, developers can enhance their software development quality.
- Project Metrics:** Project Metrics determine the overall quality of a project. It is used to calculate costs, productivity, defects and estimate the resource and deliverables of a project.

It is incredibly vital to identify the correct testing metrics for the process. Few factors to consider-

- Choose your target audiences wisely before preparing the metrics
  - Define the goal behind designing the metrics
  - Prepare metrics by considering the specific requirements of the project
  - Evaluate the financial gain behind each metrics
  - Pair the metrics with the project lifestyle phase that delivers optimum output
- Software testing can be further divided into manual and automated testing.

In manual testing, the test is performed by QA analysts in a step-by-step process. Meanwhile, in automated testing, tests are executed with the help of test automation frameworks, tools, and software.

Both manual and automated testing has its strength and weakness.

Manual testing is a slow process, but it allows testers to handle complex scenarios.

The most significant advantage of automated testing is that it enables testers to run more testing in less time, covering a substantial level of permutations, which is nearly impossible to calculate manually.

#### Types of Manual Test Metrics

Manual Test Metrics are of two types-

##### Base Metrics

Base metrics are data collected by analysts during test case development and execution. These metrics are submitted to test leads and project managers by preparing a project status report. It is quantified by using calculated metrics –

- Number of test cases
- Number of test cases executed

##### Calculated Metrics

Calculated metrics are derived using data from base metrics. The test lead gathers these data and converts them to more meaningful information for tracking the progress of projects at the module level, tester level, etc.

It comprises a significant part of SDLC and empowers developers to make vital improvements in software.

#### Most used Metrics

Below are the types of metrics, popularly used by developers and testers

- **Defect metrics:** This metric allows developers to understand the various quality aspects of software, including functionality, performance, installation stability, usability, compatibility, etc.
- **Defects finding rate:** It is used to identify the pattern of defects during a specific timeframe
- **Defect severity:** It enables the developer to understand how the defect is going to impact the quality of the software.
- **Defect cause:** It is used to understand the root cause of the defect.
- **Test Coverage:** It defines how many test cases are assigned to the program. This metric ensures the testing is conducted to its full completion. It further aids in checking the code flow and test functionalities.
- **Defect fixing time:** It determines the amount of time it takes to resolve a defect
- **Test case efficiency:** It tells the efficiency rate of test cases in finding defects
- **Schedule adherence:** Its primary motive is to figure out the time difference between the planned schedule and the actual time of executing a schedule.

#### Test Metrics Life Cycle

The life cycle of test metrics consists of four stages-

- **Analysis:** In this stage, developers identify the required metrics and define them.
- **Communicate:** Once metrics are identified, developers have to explain their importance to stakeholders and the testing team.
- **Evaluation:** This stage includes quantifying and verifying the data. Then testers have to use the data to calculate the value of the metric.
- **Report:** Once the evaluation process is finished, the development team needs to create a report including a detailed summary of the conclusion. Then the report is distributed among stakeholders and relevant representatives. The stakeholders then give their feedback after reading the information carefully.

Metrics and statistics play a vital role in measuring and evaluating various aspects of performance, progress, and outcomes in different domains. They provide objective and quantifiable data that can be used to assess the effectiveness, efficiency, and quality of processes, systems, or initiatives. Here are some common metrics and statistics used in different contexts:

**Key Performance Indicators (KPIs):** KPIs are specific metrics that organizations use to measure progress toward their goals and objectives. KPIs vary depending on the industry and the specific objectives being measured. Examples include sales revenue, customer satisfaction scores, website traffic, employee turnover rate, and production efficiency.

**Quality Metrics:** These metrics assess the quality of products or services. They can include defect rates, customer complaints, return rates, and customer satisfaction ratings. Quality metrics help organizations identify areas for improvement and track their progress in delivering high-quality offerings.

**Financial Metrics:** Financial metrics measure the financial health and performance of a business or organization. Examples include revenue growth, profit margins, return on investment (ROI), cash flow, and debt-to-equity ratio. Financial metrics help evaluate profitability, liquidity, and overall financial sustainability.

**Customer Metrics:** These metrics focus on understanding and evaluating the customer experience and satisfaction. Examples include Net Promoter Score (NPS), customer retention rate, customer lifetime value, and customer complaints. Customer metrics provide insights into customer loyalty, preferences, and perceptions.

**Productivity Metrics:** Productivity metrics assess the efficiency and output of individuals, teams, or processes. Examples include units produced per hour, average handling time for customer inquiries, response time to support tickets, and employee utilization rates. Productivity metrics help identify bottlenecks, optimize resource allocation, and improve overall efficiency.

**User Engagement Metrics:** These metrics are used in digital and online contexts to assess user engagement and interaction. Examples include website traffic, page views, bounce rate, click-through rate, time spent on a page, and social media engagement (likes, shares, comments). User engagement metrics help evaluate the effectiveness of digital marketing efforts, user experience, and content performance.

**Risk and Compliance Metrics:** Risk and compliance metrics assess an organization's adherence to regulatory requirements, internal policies, and risk management practices. Examples include compliance violation incidents, risk exposure levels, audit findings, and cybersecurity incidents. These metrics help organizations monitor their risk posture and ensure compliance with legal and regulatory obligations.

**Process Efficiency Metrics:** These metrics evaluate the efficiency and effectiveness of specific processes within an organization. Examples include cycle time, throughput, defect rate, and rework rate. Process efficiency metrics help identify areas of improvement, streamline operations, and reduce waste.

**Market and Industry Statistics:** These statistics provide insights into market trends, customer demographics, industry benchmarks, and competitive landscape. Examples include market size, market share, growth rates, customer segmentation data, and industry-specific performance indicators. Market and industry

It's important to note that the selection of metrics and statistics should align with the specific objectives, context, and requirements of the organization or domain being measured. They should be relevant, measurable, and provide actionable insights to drive improvement and informed decision-making.

### UNIT III TEST DESIGN AND EXECUTION

6

Test Objective Identification, Test Design Factors, Requirement identification, Testable Requirements, Modeling a Test Design Process, Modeling Test Results, Boundary Value Testing, Equivalence Class Testing, Path Testing, Data Flow Testing, Test Design Preparedness Metrics, Test Case Design Effectiveness, Model-Driven Test Design, Test Procedures, Test Case Organization and Tracking, Bug Reporting, Bug LifeCycle.

#### Test Objective Identification

#### Test Design Factors

#### Introduction

For designing Test Cases the following factors are considered:

1. Correctness
2. Negative
3. User Interface
4. Usability
5. Performance
6. Security
7. Integration
8. Reliability
9. Compatibility

**Correctness :** Correctness is the minimum requirement of software, the essential purpose of testing. The tester may or may not know the inside details of the software module under test e.g. control flow, data flow etc.

**Negative :** In this factor we can check what the product it is not supposed to do.

**User Interface :** In UI testing we check the user interfaces. For example in a web page we may check for a button. In this we check for button size and shape. We can also check the navigation links.

**Usability :** Usability testing measures the suitability of the software for its users, and is

directed at measuring the following factors with which specified users can achieve specified goals in particular environments.

1. **Effectiveness** : The capability of the software product to enable users to achieve specified goals with the accuracy and completeness in a specified context of use.
2. **Efficiency** : The capability of the product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

**Performance** : In software engineering, performance testing is testing that is performed from one perspective to determine how fast some aspect of a system performs under a particular workload.

Performance testing can serve various purposes. It can demonstrate that the system needs performance criteria.

1. **Load Testing**: This is the simplest form of performance testing. A load test is usually conducted to understand the behavior of the application under a specific expected load.
2. **Stress Testing**: Stress testing focuses on the ability of a system to handle loads beyond maximum capacity. System performance should degrade slowly and predictably without failure as stress levels are increased.
3. **Volume Testing**: Volume testing belongs to the group of non-functional values tests. Volume testing refers to testing a software application for a certain data volume. This volume can in generic terms be the database size or it could also be the size of an interface file that is the subject of volume testing.

**Security** : Process to determine that an Information System protects data and maintains functionality as intended. The basic security concepts that need to be covered by security testing are the following:

1. **Confidentiality** : A security measure which protects against the disclosure of information to parties other than the intended recipient that is by no means the only way of ensuring
2. **Integrity**: A measure intended to allow the receiver to determine that the information which it receives has not been altered in transit other than by the originator of the information.
3. **Authentication**: A measure designed to establish the validity of a transmission, message or originator. Allows a receiver to have confidence that the information it receives originated from a specific known source.
4. **Authorization**: The process of determining that a requester is allowed to receive a service/perform an operation.

**Integration :** Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested.

**Reliability :** Reliability testing is to monitor a statistical measure of software maturity over time and compare this to a desired reliability goal.

**Compatibility :** Compatibility testing of a part of software's non-functional tests. This testing is conducted on the application to evaluate the application's compatibility with the computing environment. Browser compatibility testing can be more appropriately referred to as user experience testing. This requires that the web applications are tested on various web browsers to ensure the following:

Users have the same visual experience irrespective of the browsers through which they view the web application.

In terms of functionality , the application must behave and respond the same across various browsers.

## Requirement identification

Requirements analysis (requirements engineering) is the process of determining user expectations for a new or modified product. It is usually a team effort and demands a variety of human soft skills, such as critical thinking, communication and judgment.

In requirements-based testing, test cases are designed based on test objectives and test conditions that are derived from requirements. Examples of requirements-based tests are those that exercise specific functions or probe nonfunctional attributes, such as reliability or usability.

Requirement identification is a crucial step in the software development and project management process. It involves identifying, analyzing, and documenting the needs, expectations, and specifications of stakeholders for a software system or solution. The goal is to gather a comprehensive and accurate understanding of what the system should accomplish and how it should function. Here are some key steps in requirement identification:

Stakeholder identification: Identify all the individuals, groups, or entities that have a vested interest in the software system. This may include end-users, customers, business owners, subject matter experts, developers, and other relevant stakeholders.

Requirements gathering techniques: Employ various techniques to gather requirements effectively. Common techniques include interviews, surveys, workshops, brainstorming sessions, observation, document analysis, and prototyping. Each technique helps to extract information from stakeholders and capture their expectations and needs.

Elicitation of requirements: Engage with stakeholders to elicit their requirements. This involves asking open-ended questions, actively listening, and encouraging stakeholders to express their needs, concerns, and desired functionalities of the system. Use the chosen requirements gathering techniques to facilitate this process.

Requirement documentation: Document the gathered requirements in a structured manner to ensure clarity, consistency, and traceability. This typically involves creating requirement documents such as a Requirements Specification Document (RSD) or User Stories, which capture the functional, non-functional, and user-related requirements.

Requirement analysis and prioritization: Analyze the gathered requirements to identify any conflicts, redundancies, or gaps. Prioritize requirements based on their importance, urgency, and alignment with the project goals. This helps in allocating resources and determining the order in which the requirements will be addressed.

Requirement validation: Validate the requirements with stakeholders to ensure that they accurately reflect their needs and expectations. This can involve conducting review sessions, obtaining feedback, and seeking clarification on any ambiguous or conflicting requirements.

Requirement management: Establish a system for managing requirements throughout the software development lifecycle. This includes tracking changes, maintaining a version history, and ensuring that requirements remain consistent and up-to-date as the project progresses.

Requirement traceability: Establish traceability between requirements and other project artifacts, such as design documents, test cases, and code. This helps ensure that each requirement is adequately addressed and tested during the development process.

Requirement communication and collaboration: Foster clear and effective communication with stakeholders throughout the requirement identification process. Collaboration tools, documentation repositories, and regular meetings can facilitate effective communication and ensure that stakeholders remain informed and involved.

Requirement validation and sign-off: Seek formal validation and sign-off from stakeholders to confirm that the identified requirements meet their expectations and needs. This provides a foundation for the subsequent stages of design, development, and testing.

Requirement identification is an iterative process that may involve multiple rounds of gathering, analysis, and validation. Effective requirement identification sets the stage for a successful software development project by ensuring a shared understanding of the desired system and facilitating accurate planning, design, and implementation.

## **Testable Requirements**

A testable requirement is one that's expressed so clearly, so unambiguously, so completely, that there can only be one interpretation of what's actually required, and from which test cases could be designed which would demonstrate clearly, unambiguously, and (just as important) cost-effectively whether the requirement ...

Performance Requirement: The system shall respond to user input within 2 seconds, measured from the time the input is entered to the time the system provides a response.

Test: Measure the response time of the system for different user inputs and verify that it meets the requirement of responding within 2 seconds.

Security Requirement: The system shall enforce user authentication by requiring a valid username and password combination.

Test: Attempt to log in with invalid or incorrect credentials and verify that the system denies access. Then, log in with valid credentials and verify that the system grants access.

**Usability Requirement:** The system shall provide a user interface that is intuitive and easy to navigate.

**Test:** Conduct user testing sessions where participants are given tasks to perform using the system. Observe their interactions and gather feedback to determine if the interface meets the requirement of being intuitive and easy to navigate.

**Reliability Requirement:** The system shall have an uptime of at least 99% over a 30-day period.

**Test:** Monitor the system's availability and record any instances of downtime. Calculate the uptime percentage over a 30-day period and verify that it meets or exceeds the requirement of 99%.

**Compatibility Requirement:** The system shall be compatible with the latest versions of popular web browsers (e.g., Chrome, Firefox, Safari).

**Test:** Install and run the system on different web browsers' latest versions, ensuring that all functionalities work as expected without any major issues or compatibility errors.

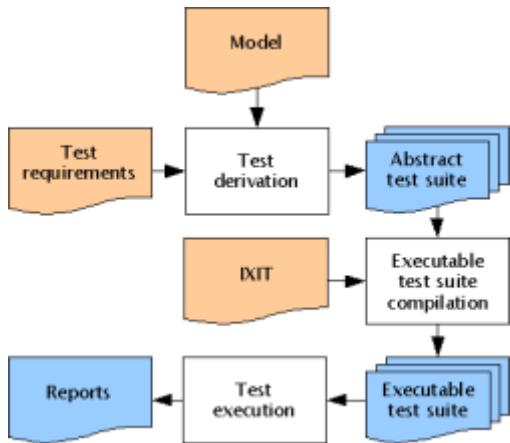
**Accuracy Requirement:** The system shall calculate mathematical equations with an accuracy of at least 95%.

**Test:** Create a set of mathematical equations with known solutions and input them into the system. Compare the system's calculated results with the expected results and verify that the accuracy is at least 95%.

## **Modeling a Test Design Process**

Model-based testing is an application of model-based design for designing and optionally also executing artifacts to perform software testing or system testing. Models can

be used to represent the desired behavior of a system under test (SUT), or to represent testing strategies and a test environment.



**Define Test Objectives:** Start by clearly defining the objectives of the testing process. Determine what you want to achieve through testing, such as identifying defects, ensuring system functionality, or validating requirements.

**Identify Test Requirements:** Identify the specific requirements that need to be tested. This includes functional requirements, non-functional requirements (e.g., performance, security), and any other relevant criteria.

**Create a Test Strategy:** Develop a high-level test strategy that outlines the overall approach to testing. Define factors like test levels (e.g., unit testing, integration testing, system testing), test types (e.g., functional testing, performance testing), and the overall test environment.

**Develop Test Plan:** Based on the test strategy, create a detailed test plan. The test plan should include specific test objectives, scope, test deliverables, entry/exit criteria, test schedule, resource allocation, and any dependencies.

**Define Test Cases:** Identify and document individual test cases that will be executed during testing. Test cases should cover various scenarios and conditions to thoroughly validate the system. Each test case should have a unique identifier, preconditions, test steps, expected results, and any additional information or attachments.

**Prioritize Test Cases:** Prioritize test cases based on factors such as criticality, risk, and frequency of use. This helps ensure that the most critical and high-impact tests are executed first.

**Design Test Data:** Identify and create the necessary test data required to execute the test cases effectively. Test data should cover a range of scenarios and conditions, including valid and invalid inputs, boundary values, and edge cases.

**Prepare Test Environment:** Set up the test environment, including hardware, software, and any required test tools. Ensure that the environment closely matches the production environment to ensure accurate testing.

**Execute Test Cases:** Execute the defined test cases as per the test plan. Record the actual results, compare them against the expected results, and document any deviations or defects found during testing.

**Track and Manage Defects:** Use a defect tracking system to log and track any issues or defects identified during testing. Assign priorities and severities to each defect and ensure they are resolved and retested before the final product release.

**Review and Evaluate:** Review the test results, metrics, and any other relevant data to assess the overall quality of the system under test. Identify areas for improvement and make necessary adjustments to the test process.

**Report and Communicate:** Prepare a comprehensive test report summarizing the test activities, findings, and recommendations. Communicate the results to the stakeholders, including developers, project managers, and other relevant parties.

### **Modeling Test Results**

Model-based testing is a software testing technique where the run time behavior of the software under test is checked against predictions made by a model. A model is a description of a system's behavior. Behavior can be described in terms of input sequences, actions, conditions, output, and flow of data from input to output. It should be practically understandable and can be reusable; shareable must have a precise description of the system under test.

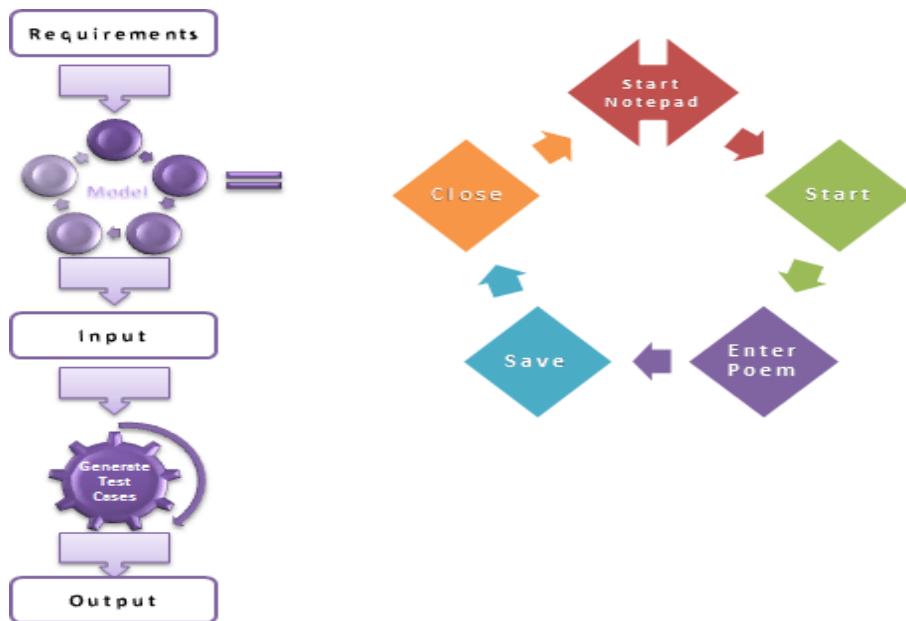
There are numerous models available, and it describes different aspects of the system behavior. Examples of the model are:

- Data Flow
- Control Flow
- Dependency Graphs
- Decision Tables
- State transition machines

Model-Based Testing describes how a system behaves in response to an action (determined by a model). Supply action, and see if the system responds as per the expectation.

It is a lightweight formal method to validate a system. This testing can be applied to both hardware and software testing.

## Model Based Testing Example



The above model explains the simplified approach of writing poetry in notepad and possible actions related to each step. For each and every action (like starting, Entering a poem, Saving), [Test Case](#) can be generated, and the output can be verified.

### Types of MBT

There are two types of Model based testing frameworks-

1. Offline / a priori: Generation of Test Suites before executing it. A test suite is nothing but a collection of test cases.
2. Online / on-the-fly: Generation of Test Suites during test execution

### Different Models in Testing

In order to understand the MBT, it is necessary to understand some of the models explained below. Let's go through them one by one:

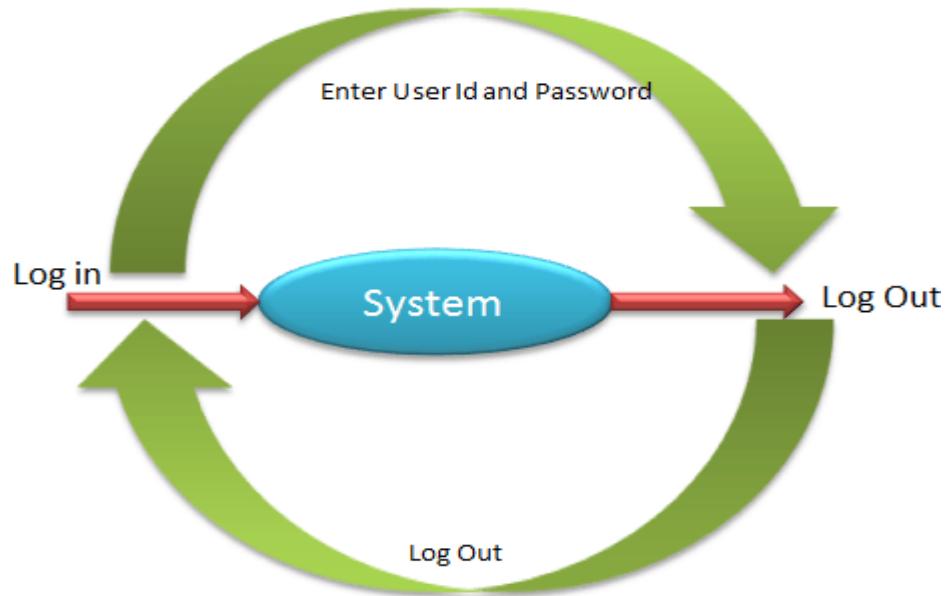
#### Finite State Machines

This model helps testers to assess the result depending on the input selected. Various combinations of the inputs can result in a corresponding state of the system.

The system will have a specific state and current state, which is governed by a set of inputs given by the testers.

Consider the example-

There is a system that allows employees to log- into the application. Now, the current state of the employee is ||Out|| and it became ||In|| once he signs- into the system. Under the ||in|| state, an employee can view, print, and scan documents in the system.

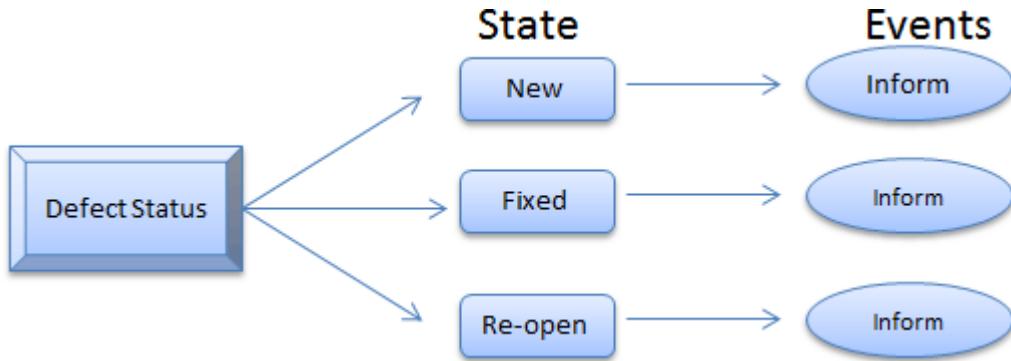


### State Charts

It is an extension of the Finite state machine and can be used for complex and real time systems. Statecharts are used to describe various behaviors of the system. It has a definite number of states. The behavior of the system is analyzed and represented in the form of events for each state.

#### For example –

Defects are raised in the defect management tool with the status as New. Once it is fixed by developers, it has to be changed to status Fixed. If a defect is not fixed, change status to Re-open. State charts should be designed in such a way that it should call for an event for each state.

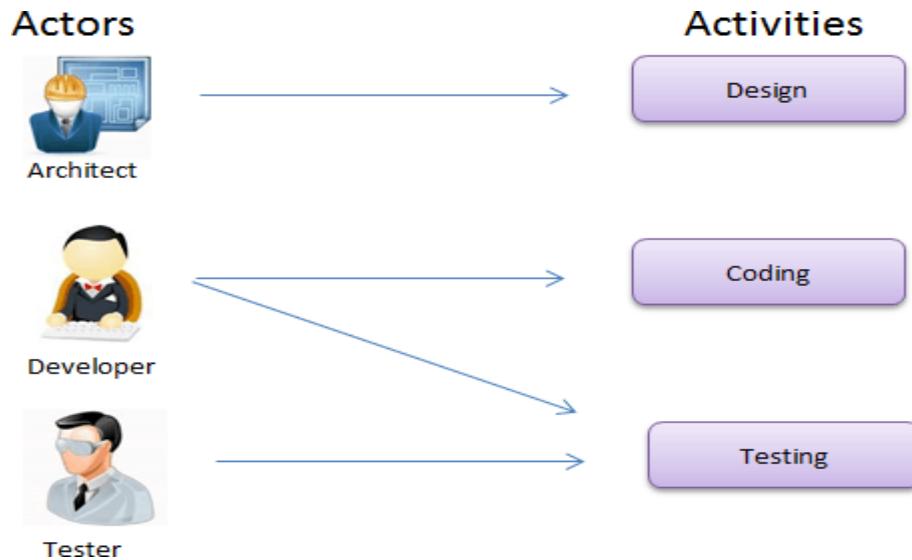


### Unified Modeling Language (UML)

**Unified Modeling Language (UML)** is a standardized general-purpose modeling language. UML includes a set of graphic notation techniques to create visual models that can describe the very complicated behavior of the system.

**UML has notations such as:**

- Activities
- Actors
- Business Process
- Components
- Programming language



### Challenges of Model Based Testing

Deployment of MBT in every organization obviously requires a high amount of investment and effort. Following are drawbacks of MBT in Software Engineering.

- Necessary Skills required in testers
- Learning curve time will be more
- Difficult to understand the model itself

### Advantages of Model Testing

The following are benefits of MBT:

- Easy test case/suite maintenance
- Reduction in Cost
- Improved Test Coverage
- Can run different tests on n number of machines
- Early defect detection
- Increase in defect count

- Time savings
- Improved tester job satisfaction

## Boundary Value Testing

Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

|                |                     |
|----------------|---------------------|
| <b>Name</b>    | Enter Your Name     |
| <b>Age</b>     | Between 18 to 30    |
| <b>Adhar</b>   | Number of 12 Digits |
| <b>Address</b> | Enter Your Address  |

Testing of boundary values is done by making valid and invalid partitions. Invalid partitions are tested because testing of output in adverse condition is also essential.

### Let's understand via practical:

Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists

of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



| Invalid test cases     | Valid test cases       | Invalid test cases     |
|------------------------|------------------------|------------------------|
| 11, 13, 14, 15, 16, 17 | 18, 19, 24, 27, 28, 30 | 31, 32, 36, 37, 38, 39 |

The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error message.

If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects.

### Equivalence Class Testing

Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as other.

The equivalence partitions are derived from requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.

### Examples of Equivalence Partitioning technique

Assume that there is a function of a software application that accepts a particular number of digits, not greater and less than that particular number. For example, an OTP

number which contains only six digits, less or more than six digits will not be accepted, and the application will redirect the user to the error page.

1. 1. OTP Number = 6 digits

| INVALID     | INVALID     | VALID       | VALID      |
|-------------|-------------|-------------|------------|
| 1 Test case | 2 Test case | 3 Test case |            |
| DIGITS >=7  | DIGITS<=5   | DIGITS = 6  | DIGITS = 6 |

93847262      9845      456234      451483

1. 2. Mobile number = 10

digits

| <b>INVALID</b><br>1 Test case | <b>INVALID</b><br>2 Test case | <b>VALID</b><br>3 Test case | <b>VALID</b> |
|-------------------------------|-------------------------------|-----------------------------|--------------|
| DIGITS >=11                   | DIGITS<=9                     | DIGITS = 10                 | DIGITS =10   |
| 93847262219                   | 984543985                     | 9991456234                  | 9893451483   |

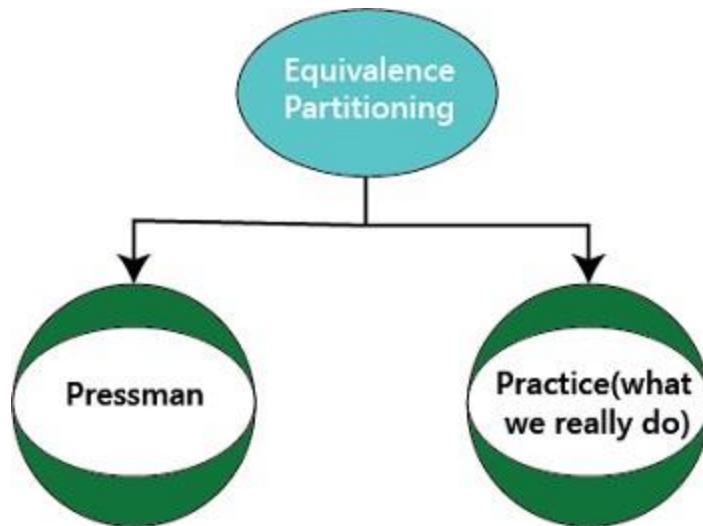
In both examples, we can see that there is a partition of two equally valid and invalid partitions, on applying valid value such as OTP of six digits in the first example and mobile number of 10 digits in the second example, both valid partitions behave same, i.e. redirected to the next page.

Another two partitions contain invalid values such as 5 or less than 5 and 7 or more than 7 digits in the first example and 9 or less than 9 and 11 or more than 11 digits in the second example, and on applying these invalid values, both invalid partitions behave same, i.e. redirected to the error page.

We can see in the example, there are only three test cases for each example and that is also the principle of equivalence partitioning which states that this method intended to reduce the number of test cases.

### How we perform equivalence partitioning

We can perform equivalence partitioning in two ways which are as follows:



Let us see how pressman and general practice approaches are going to use in different conditions:

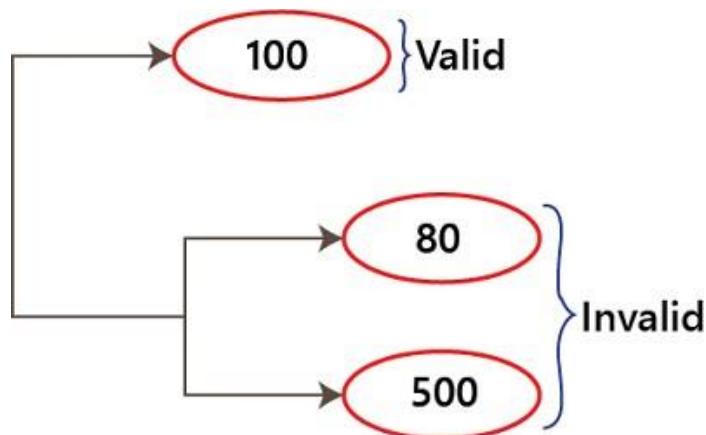
### Condition1

If the requirement is **a range of values**, then derive the test case for **one valid and two invalid** inputs.

Here, the **Range of values** implies that whenever we want to identify the range values, we go for equivalence partitioning to achieve the minimum test coverage. And after that, we go for error guessing to achieve maximum test coverage.

#### According to pressman:

For example, the Amount of test field accepts a Range (100-400) of values:

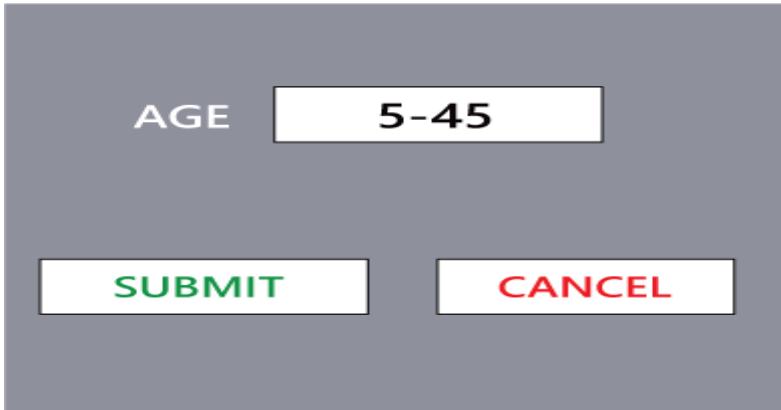


#### According to General Practice method:

Whenever the requirement is Range + criteria, then divide the Range into the internals and check for all these values.

#### For example:

In the below image, the pressman technique is enough to test for an age text field for one valid and two invalids. But, if we have the condition for insurance of ten years and above are required and multiple policies for various age groups in the age text field, then we need to use the practice method.



### Condition2

If the requirement is a **set of values**, then derive the test case for **one valid** and **two invalid** inputs.

Here, **Set of values** implies that whenever we have to test a set of values, we go for **one positive** and **two negative** inputs, then we move for error guessing, and we also need to verify that all the sets of values are as per the requirement.

### Example 1

#### Based on the Pressman Method

If the Amount Transfer is (100000-700000)

Then for, 1 lakh →Accept

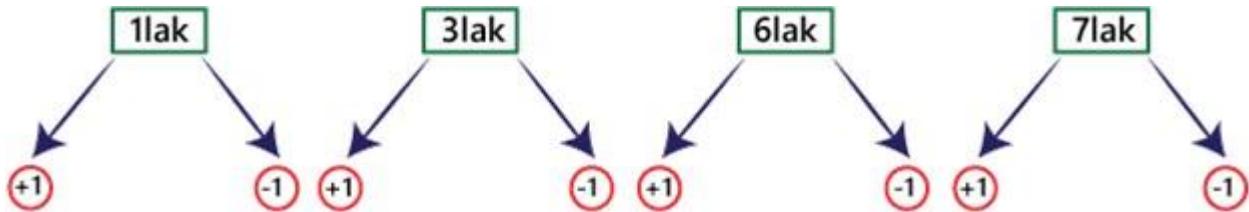
#### And according to General Practice method

The Range + Percentage given to 1 lakh - 7 lakh

**Like:** 1lak - 3lak →5.60%

3lak - 6lak →3.66%

6lak - 7lak →Free



If we have things like loans, we should go for the general practice approach and separate the stuff into the intervals to achieve the minimum test coverage.

### Example 2

if we are doing online shopping, mobile phone product, and the different **Product ID** - 1,4,7,9

**ONLINE SHOPING**

---

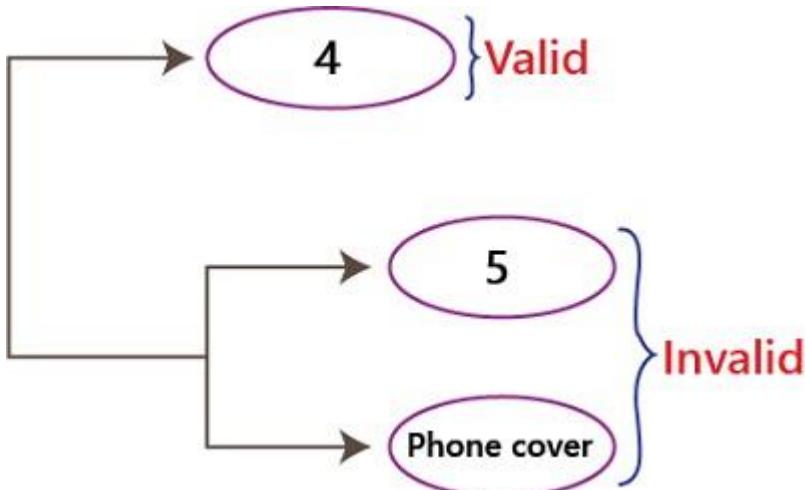
Product ID

---

**SUBMIT**      **CANCEL**

Here, 1 → phone covers 4 → earphones 7 → charger 9 → Screen guard

And if we give the product id as 4, it will be accepted, and it is one valid value, and if we provide the product id as 5 and phone cover, it will not be accepted as per the requirement, and these are the two invalid values.



### Condition 3

If the requirement is **Boolean (true/false)**, then derive the test case for both true/false values.

The Boolean value can be true and false for the radio button, checkboxes.

**For example**

A user interface mockup showing a form for gender selection. The form contains two radio buttons: one labeled "Female" and another labeled "Male". At the bottom of the form are two buttons: "SUBMIT" on the left and "CANCEL" on the right.

| Serial no | Description | Input | Expected | Note |
|-----------|-------------|-------|----------|------|
|           |             |       |          |      |

|   |                |    |                                                           |                                                          |
|---|----------------|----|-----------------------------------------------------------|----------------------------------------------------------|
| 1 | Select valid   | NA | True                                                      | ---                                                      |
| 2 | Select invalid | NA | False                                                     | Values can be change based according to the requirement. |
| 3 | Do not select  | NA | Do not select anything, error message should be displayed | We cannot go for next question                           |
| 4 | Select both    | NA | We can select any radio button                            | Only one radio button can be selected at a time.         |

**Note:**

In **Practice** method, we will follow the below process:

Here, we are testing the application by deriving the below inputs values:

|    |     |      |      |      |      |      |      |      |
|----|-----|------|------|------|------|------|------|------|
| 80 | 500 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
|----|-----|------|------|------|------|------|------|------|

Let us see one program for our better understanding.

1. If( amount < 500 or > 7000)
2. {
3. Error Message
4. }
5. if( amount is between 500 & 3000)
6. {
7. deduct 2%
8. }

```

9. if (amount > 3000)
10. {
11. deduct 3%
12. }

```

When the pressman technique is used, the first two conditions are tested, but if we use the practice method, all three conditions are covered.

We don't need to use the practice approach for all applications. Sometime we will use the pressman method also.

But, if the application has much precision, then we go for the practice method.

If we want to use the practice method, it should follow the below aspects:

- It should be product-specific
- It should be case-specific
- The number of divisions depends on the precision( 2% and 3 % deduction)

### Advantages and disadvantages of Equivalence Partitioning technique

Following are pros and cons of equivalence partitioning technique:

| <b>Advantages</b>                                                                          | <b>disadvantages</b>                                                                                                             |
|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| It is process-oriented                                                                     | All necessary inputs may not cover.                                                                                              |
| We can achieve the Minimum test coverage                                                   | This technique will not consider the condition for boundary value analysis.                                                      |
| It helps to decrease the general test execution time and also reduce the set of test data. | The test engineer might assume that the output for all data set is right, which leads to the problem during the testing process. |

## Path Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the **black box testing** and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

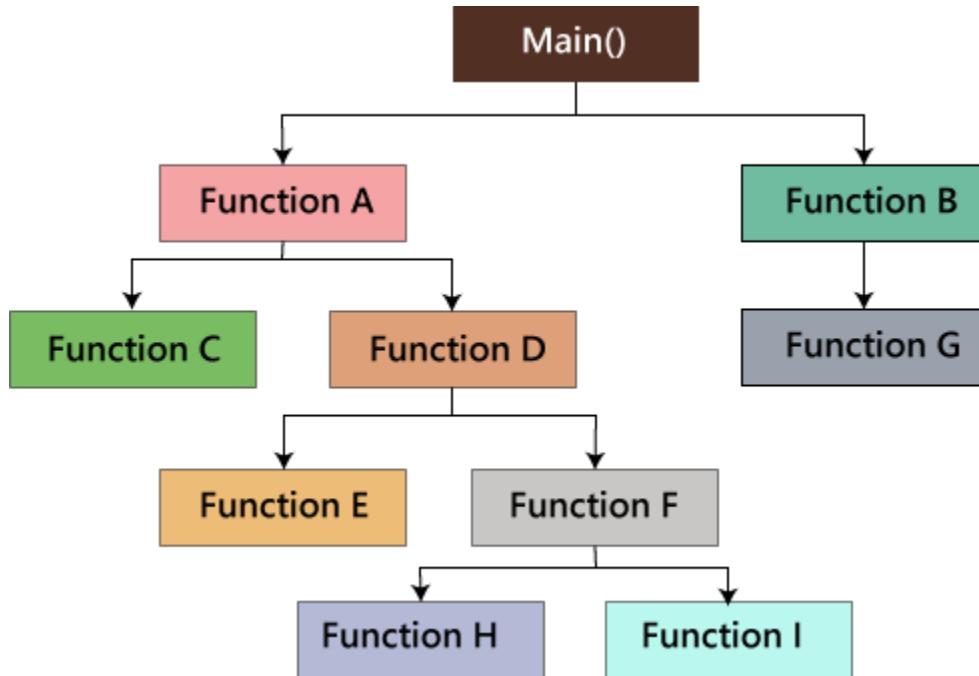
- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program

### Path testing

In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



And test all the independent paths implies that suppose a path from main() to function G, first set the parameters and test if the program is correct in that particular path, and in the same way test all other paths and fix the bugs.

### Data Flow Testing

**Data Flow Testing** is a type of structural testing. It is a method that is used to find the test paths of a program according to the locations of definitions and uses of variables in the program. It has nothing to do with data flow diagrams.

It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

To illustrate the approach of data flow testing, assume that each statement in the program assigned a unique statement number. For a statement number S-

$\text{DEF}(S) = \{X \mid \text{statement } S \text{ contains the definition of } X\}$

$\text{USE}(S) = \{X \mid \text{statement } S \text{ contains the use of } X\}$

If a statement is a loop or if condition then its DEF set is empty and USE set is based on the condition of statement s.

Data Flow Testing uses the control flow graph to find the situations that can interrupt the flow of the program.

Reference or define anomalies in the flow of the data are detected at the time of associations between values and variables. These anomalies are:

- A variable is defined but not used or referenced,
- A variable is used but never defined,
- A variable is defined twice before it is used

### **Advantages of Data Flow Testing:**

Data Flow Testing is used to find the following issues-

- To find a variable that is used but never defined,
- To find a variable that is defined but never used,
- To find a variable that is defined multiple times before it is used,
- Deallocating a variable before it is used.

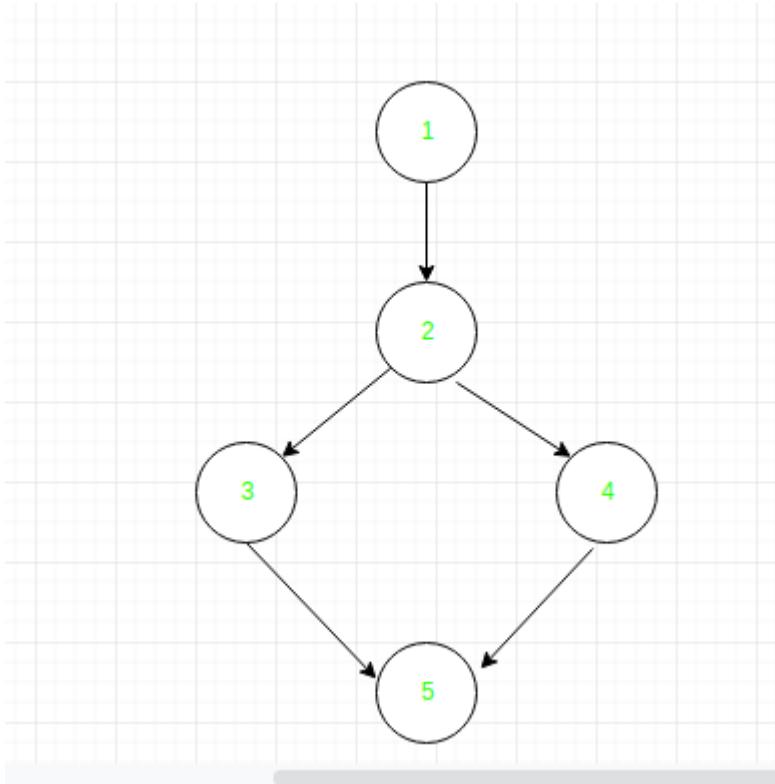
### **Disadvantages of Data Flow Testing**

- Time consuming and costly process
- Requires knowledge of programming languages

### **Example:**

1. read x, y;
2. if( $x > y$ )
3. a =  $x + 1$
- else
4. a =  $y - 1$
5. print a;

### **Control flow graph of above example:**



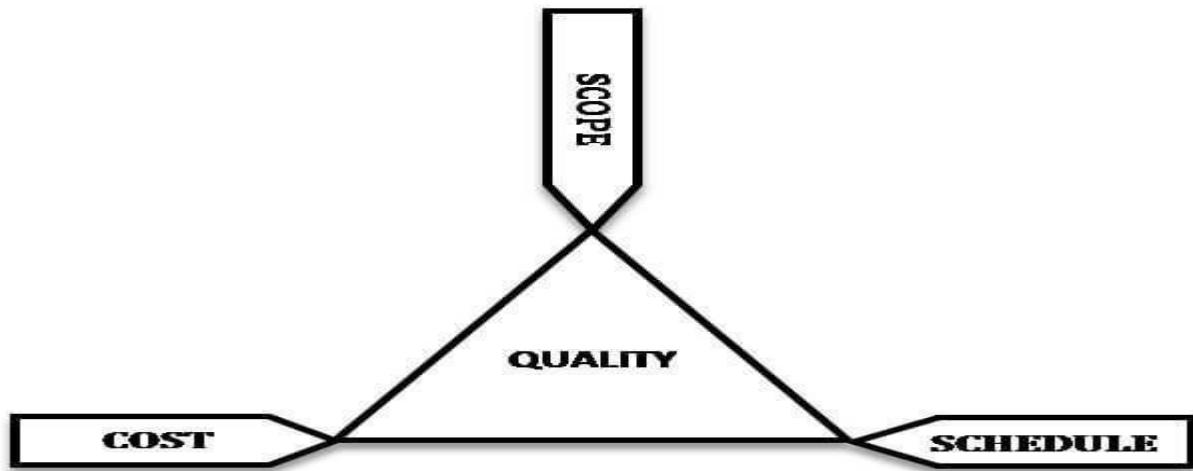
Define/use of variables of above example:

| Variable | Defined at node | Used at node |
|----------|-----------------|--------------|
| x        | 1               | 2, 3         |
| y        | 1               | 2, 4         |
| a        | 3, 4            | 5            |

### Test Design Preparedness Metrics

**Software Testing Metrics** are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.



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

### Types of Test Metrics



- **Process Metrics:** It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle)
- **Product Metrics:** It deals with the quality of the software product
- **Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

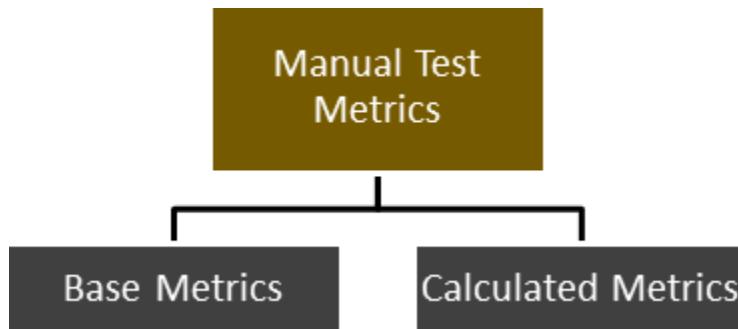
Identification of correct testing metrics is very important. Few things need to be considered before identifying the test metrics

- Fix the target audience for the metric preparation
- Define the goal for metrics
- Introduce all the relevant metrics based on project needs
- Analyze the cost benefits aspect of each metrics and the project lifestyle phase inwhich it results in the maximum output

### Manual Test Metrics

In Software Engineering, Manual test metrics are classified into two classes

- Base Metrics**
- Calculated Metrics**

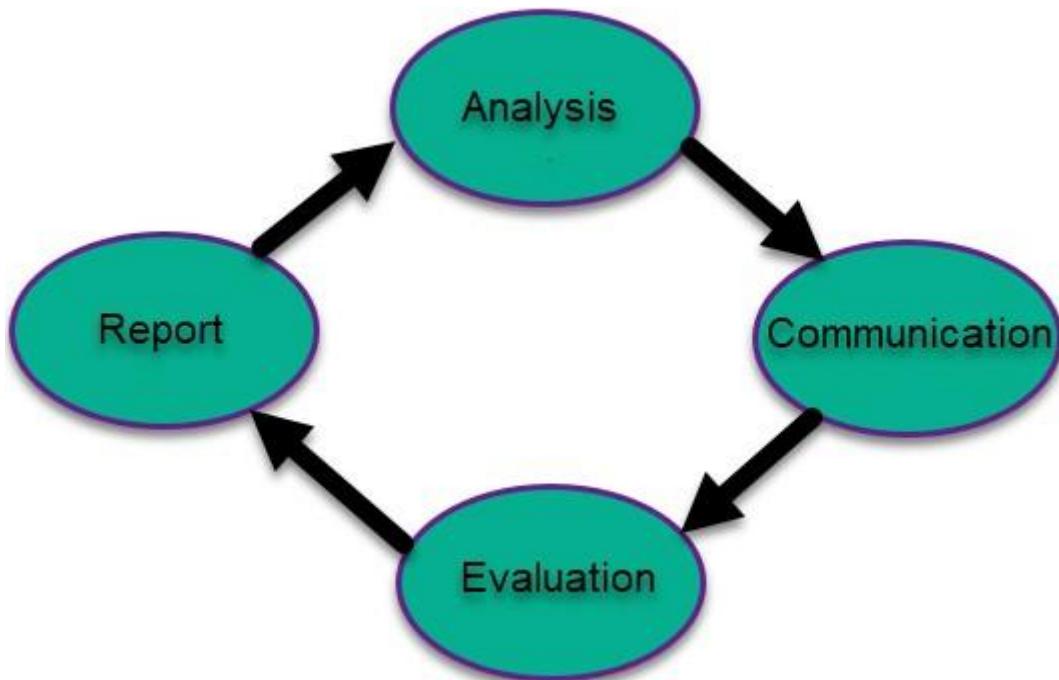


Base metrics is the raw data collected by Test Analyst during the test case developmentand execution (**# of test cases executed, # of test cases**). While calculated metrics are derived from the data collected in base metrics. Calculated metrics is usually followed by the test manager for test reporting purpose (**% Complete, % Test Coverage**).

Depending on the project or business model some of the important metrics are

- Test case execution productivity metrics
- Test case preparation productivity metrics
- Defect metrics
- Defects by priority
- Defects by severity
- Defect slippage ratio

### Test Metrics Life Cycle in Software Engineering



### Different stages of Metrics life cycle

### Steps during each stage

|             |                                                                                                                                                                                                                                     |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Analysis    | <ol style="list-style-type: none"> <li>1. Identification of the Metrics</li> <li>2. Define the identified QA Metrics</li> </ol>                                                                                                     |
| Communicate | <ol style="list-style-type: none"> <li>1. Explain the need for metric to stakeholder and testing team</li> <li>2. Educate the testing team about the data points to need to be capturing while processing the metric</li> </ol>     |
| Evaluation  | <ol style="list-style-type: none"> <li>1. Capture and verify the data</li> <li>2. Calculating the metrics value using the data captured</li> </ol>                                                                                  |
| Report      | <ol style="list-style-type: none"> <li>1. Develop the report with an effective conclusion</li> <li>2. Distribute the report to the stakeholder and respective representatives</li> <li>3. Take feedback from stakeholder</li> </ol> |

### How to calculate Test Metric

| Sr# | Steps to test metrics                                      | Example                           |
|-----|------------------------------------------------------------|-----------------------------------|
| 1   | Identify the key software testing processes to be measured | Testing progress tracking process |

|   |                                                                                                     |                                                                                                                              |
|---|-----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| 2 | In this Step, the tester uses the data as a baseline to define the metrics                          | The number of test cases planned to be executed per day                                                                      |
| 3 | Determination of the information to be followed, a frequency of tracking and the person responsible | The actual test execution per day will be captured by the manager at the end of the day                                      |
| 4 | Effective calculation, management, and interpretation of the defined metrics                        | The actual test cases executed per day                                                                                       |
| 5 | Identify the areas of improvement depending on the interpretation of defined metrics                | The <u>Test Case</u> execution falls below the goal set, we need investigate the reason and suggest the improvement measures |

### Example of Test Metric

To understand how to calculate the test metrics, we will see an example of a percentage test case executed.

To obtain the execution status of the test cases in percentage, we use the formula.

Percentage test cases executed= (No of test cases executed/ Total no of test cases written) X 100

Likewise, you can calculate for other parameters like **test cases not executed, test cases passed, test cases failed, test cases blocked, etc.**

### Test Metrics Glossary

- **Rework Effort Ratio** = (Actual rework efforts spent in that phase/ total actual efforts spent in that phase) X 100
- **Requirement Creep** = ( Total number of requirements added/No of initial requirements)X100
- **Schedule Variance** = (Actual Date of Delivery - Planned Date of Delivery)
- **Cost of finding a defect in testing** = ( Total effort spent on testing/ defects found in testing)
- **Schedule slippage** = (Actual end date - Estimated end date) / (Planned End Date - Planned Start Date) X 100
- **Passed Test Cases Percentage** = (Number of Passed Tests/Total number of tests executed) X 100
- **Failed Test Cases Percentage** = (Number of Failed Tests/Total number of tests executed) X 100
- **Blocked Test Cases Percentage** = (Number of Blocked Tests/Total number of tests executed) X 100
- **Fixed Defects Percentage** = (Defects Fixed/Defects Reported) X 100

- **Accepted Defects Percentage** = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100
- **Defects Deferred Percentage** = (Defects deferred for future releases /Total Defects Reported) X 100
- **Critical Defects Percentage** = (Critical Defects / Total Defects Reported) X 100
- **Average time for a development team to repair defects** = (Total time taken for bugfixes/Number of bugs)
- **Number of tests run per time period** = Number of tests run/Total time
- **Test design efficiency** = Number of tests designed /Total time
- **Test review efficiency** = Number of tests reviewed /Total time
- **Bug find rate or Number of defects per test hour** = Total number of defects/Total number of test hours

Test design preparedness metrics are used to evaluate the readiness and effectiveness of test designs in software testing. These metrics help assess the quality and completeness of test cases, test scripts, and other test artifacts before execution. Here are some commonly used test design preparedness metrics:

**Test Case Completeness:** This metric measures the percentage of test cases that have been defined and documented compared to the total number of expected test cases. It provides insights into the extent of test coverage and helps identify any gaps or missing scenarios.

**Requirement Coverage:** This metric evaluates the percentage of requirements covered by test cases. It ensures that all specified requirements have corresponding test cases, minimizing the risk of untested functionalities.

**Test Case Effectiveness:** This metric assesses the quality of test cases in terms of their ability to detect defects. It measures the percentage of test cases that find defects during execution. Higher effectiveness indicates that the test cases are capable of identifying issues.

**Test Script Preparation Time:** This metric measures the time taken to prepare test scripts for automated testing. It helps track the efficiency of test script development, ensuring that sufficient time is allocated for creating robust and maintainable scripts.

**Test Data Readiness:** This metric evaluates the availability and quality of test data required for test execution. It assesses whether the necessary data sets are prepared, validated, and readily available, minimizing delays during testing.

**Test Environment Readiness:** This metric examines the readiness of the test environment, including hardware, software, network configurations, and any other dependencies. It ensures that the environment is properly set up and all necessary components are available for testing.

**Test Design Review Findings:** This metric captures the number and severity of issues identified during test design reviews. It helps assess the effectiveness of the review process and highlights potential areas for improvement.

**Defect Leakage:** This metric measures the percentage of defects that were missed during test design and escaped to subsequent phases or production. It helps evaluate the thoroughness of test design and identify opportunities for enhancing defect detection.

These metrics can be tracked and monitored regularly to identify trends, areas for improvement, and to ensure the readiness of test designs before execution. By using these metrics, organizations can enhance the overall effectiveness and efficiency of their testing efforts.

## Test Case Design Effectiveness

Test Case Design Effectiveness is a metric used to evaluate the quality and efficiency of test cases in detecting defects. It measures how well the test cases are designed to uncover potential issues in the system under test. Here are some factors that contribute to test case design effectiveness:

**Requirement Coverage:** Effective test cases should cover all the specified requirements of the system or application being tested. The test cases should address functional and non-functional requirements, ensuring comprehensive coverage.

**Test Case Relevance:** Test cases should be relevant to the features or functionalities being tested. They should focus on critical and high-risk areas to maximize the chances of identifying defects.

**Test Case Clarity:** Well-designed test cases should be clear and easily understandable. They should provide precise instructions on how to execute the test and what the expected results should be.

**Test Case Independence:** Test cases should be independent of each other to avoid any dependencies or sequencing issues. Each test case should be able to run and produce accurate results individually.

**Test Case Completeness:** Effective test cases should cover various scenarios and input combinations to thoroughly test the system. They should include both positive and negative test cases, boundary value tests, and any other relevant test scenarios.

**Test Case Traceability:** Test cases should be traceable back to the specific requirements they are intended to validate. This helps ensure that all requirements have corresponding test cases and provides a clear linkage between the two.

**Test Case Maintainability:** Well-designed test cases are easy to maintain and update as the system evolves. They should be designed with modularity and reusability in mind, allowing for efficient maintenance and reducing the effort required for test case maintenance.

**Test Case Prioritization:** Test cases should be prioritized based on their criticality and impact on the system. This ensures that the most important and high-risk scenarios are covered early in the testing process.

By assessing the effectiveness of test case design using these factors, organizations can improve the overall quality of their testing efforts. Regular reviews, feedback loops, and collaboration between testers, developers, and other stakeholders can help refine and enhance the test case design process.

## **Model-Driven Test Design**

MDTD is built on the idea that designers will become more effective and efficient if they can raise the level of abstraction. This approach breaks down the testing into a series of small tasks that simplify test generation. Then test designers isolate their tasks and work at a higher level of abstraction by using mathematical engineering structures to design test values independently of the details of the software or design artifacts, test automation, and Test Execution.

## Different phases in MDTD

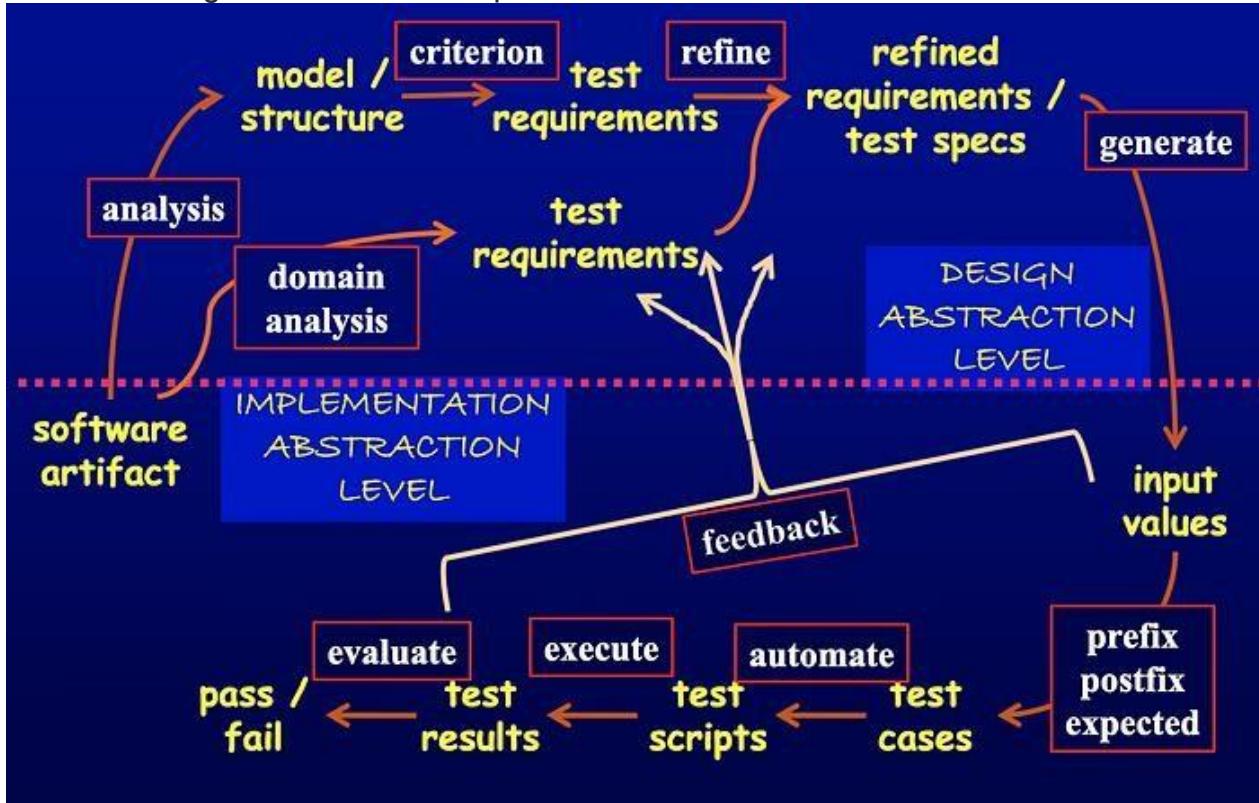
MDTD can be done in 4 different phases. Each type of activity requires different skills, background knowledge, education, and training. It is better to use different sets of people depend on the situation.

1. Test Design — This can be done in either Criteria-Based where Design test values satisfy coverage criteria or other engineering goals or in Human-Based where Design test values based on domain knowledge of the program and human knowledge of testing which is comparatively harder. This is the most technical part of the MDTD process better to use experienced developers in this phase.
2. Test Automation — This involves embedding test values to scripts. Test cases are defined based on the test requirements. Test values are chosen such that we can cover a larger part of the application with fewer test cases. We don't need that much domain knowledge in this phase, however, we need to use technically skilled people.
3. Test Execution — The test engineer will run tests and records the results in this activity. Unlike the previous activities, test execution not required a high skill set such as technical knowledge, logical thinking, or domain knowledge. Since we consider this phase comparatively low risk, we can assign junior intern engineers to

execute the process. But we should focus on monitoring, log collecting activities based on automation tools.

4. Test Evaluation — The process of evaluating the results and reporting to developers. This phase is comparatively harder and we expected to have knowledge in the domain, testing, and User interfaces, and psychology

The below diagrams shows the steps & activities involved in the MDTD



Test automation process, make it easy to do regression testing in less time compared to manual testing, and also it avoids the chance of missing the previous passed test cases

to be tested in the current testing process. MDTD defines a simple framework to automate the testing process in a structured manner.

## Test Procedures

- **Step-1: Assess Development Plan and Status –**

This initiative may be prerequisite to putting together Verification, Validation, and Testing Plan wont to evaluate implemented software solution. During this step, testers challenge completeness and correctness of event plan. Based on extensiveness and completeness of Project Plan testers can estimate quantity of resources they're going to got to test implemented software solution.

- **Step-2: Develop the Test Plan –**

Forming plan for testing will follow an equivalent pattern as any software planning process. The structure of all plans should be an equivalent, but content will vary supported degree of risk testers perceive as related to software being developed.

- **Step-3: Test Software Requirements –**

Incomplete, inaccurate, or inconsistent requirements cause most software failures. The inability to get requirement right during requirements gathering phase can also increase cost of implementation significantly. Testers, through verification, must determine that requirements are accurate, complete, and they do not conflict with another.

- **Step-4: Test Software Design –**

This step tests both external and internal design primarily through verification techniques. The testers are concerned that planning will achieve objectives of wants, also because design being effective and efficient on designated hardware.

- **Step-5: Build Phase Testing –**

The method chosen to build software from internal design document will determine type and extensiveness of testers needed. As the construction becomes more automated, less testing are going to be required during this phase. However, if software is made using waterfall process, it's subject to error and will be verified. Experience has shown that it's significantly cheaper to spot defects during development phase, than through dynamic testing during test execution step.

- **Step-6: Execute and Record Result –**

This involves testing of code during dynamic state. The approach, methods, and tools laid out in test plan are going to be wont to validate that executable code actually meets stated software requirements, and therefore the structural specifications of design.

- **Step-7: Acceptance Test –**

Acceptance testing enables users to gauge applicability and usefulness of software in performing their day-to-day job functions. This tests what user believes software should perform, as against what documented requirements state software should perform.

- **Step-8: Report Test Results –**

Test reporting is continuous process. It may be both oral and written. It is important that defects and concerns be reported to the appropriate parties as early as possible, so that corrections can be made at the lowest possible cost.

- **Step-9: The Software Installation –**

Once test team has confirmed that software is prepared for production use, power to execute that software during production environment should be tested. This tests interface to operating software, related software, and operating procedures.

- **Step-10: Test Software Changes –**

While this is often shown as Step 10, within context of performing maintenance after software is implemented, concept is additionally applicable to changes throughout implementation process. Whenever requirements changes, test plan must change, and impact of that change on software systems must be tested and evaluate.

- **Step-11: Evaluate Test Effectiveness –**

Testing improvement can best be achieved by evaluating effectiveness of testing at top of every software test assignment. While this assessment is primarily performed by testers, it should involve developers, users of software, and quality assurance professionals if function exists within the IT organization.

## Test Case Organization and Tracking

Test case organization and tracking are essential aspects of effective test management. Proper organization and tracking of test cases help ensure that testing activities are structured, well-documented, and can be easily monitored and managed. Here are some best practices for test case organization and tracking:

**Test Case Repository:** Establish a centralized repository or test management tool to store and manage test cases. This provides a single source of truth and facilitates easy access and collaboration among team members.

**Test Case Naming Convention:** Define a consistent naming convention for test cases to ensure clarity  
GOJAN SCHOOL OF BUSINESS AND TECHNOLOGY    R KANNAN AP/CSE    145

and ease of identification. Include relevant information such as the feature being tested, the test objective, and any specific identifiers.

**Test Case Structure:** Create a standardized structure for test cases that includes sections like test case ID, description, preconditions, steps, expected results, actual results, and status. This structure helps testers understand and follow the test cases consistently.

**Categorization and Classification:** Categorize test cases based on different criteria such as functionality, priority, complexity, or business process. Use labels, tags, or metadata to classify test cases accordingly, making it easier to filter and search for specific test cases.

**Test Case Hierarchy:** Organize test cases hierarchically, grouping them under appropriate modules or test suites. This helps maintain a logical and systematic arrangement, especially for large test suites.

**Test Case Versioning:** Implement version control for test cases to track changes and revisions over time. This allows for easy rollback to previous versions if needed and provides a history of modifications made to the test cases.

**Test Case Dependencies:** Identify and document dependencies between test cases to ensure proper sequencing and execution. Clearly define any prerequisite test cases or test data required for the execution of specific test cases.

**Test Case Status and Progress Tracking:** Track the status and progress of test cases throughout the testing lifecycle. Use status labels (e.g., not started, in progress, blocked, passed, failed) to indicate the current state of each test case. This provides visibility into testing progress and helps identify any bottlenecks or delays.

**Test Case Traceability:** Establish traceability between test cases and requirements or user stories. Link each test case to the specific requirement it validates, ensuring coverage and establishing a clear traceability matrix.

**Test Case Reporting:** Generate reports and metrics to monitor test case execution, pass/fail rates, and overall testing progress. Regularly review and analyze the reports to identify trends, gaps, and areas for improvement.

A test case is a series of activities and tasks that must be performed on the application to verify its workability and if it satisfies the user's requirements. Test case management involves the management of these test cases to ensure superior quality of products via efficient testing efforts.

Test case management tools are software applications that help to manage and organize the test cases, test plans, and test execution for software testing and test management.

Test case management tools help software testers manage test cases efficiently, reduce manual efforts, and ensure comprehensive test coverage.

Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain manual testing as well.

The test management tool is connected with the automation software. These types of tools had various strategies for testing and multiple sets of features. Some of the test management tools had capabilities to design the test case with the help of requirements.

It is best for test managing, scheduling, defect logging, tracking, and analysis.

Some of the most commonly used test management tools are as follows:

- **Quality center**
- **RTH**
- **Testpad**
- **Test Monitor**
- **PractiTest**



### Quality center

The quality center is a test management tool that is launched by [HP](#), which is also known as **ALM [Application life cycle management]** tool.

This is very helpful for both test management and [SDLC](#) because it supports multiple stages of the [software development life cycle](#).

It is a web-based testing tool which helps us to control the software from scratch like collection of requirement, planning, designing, testing, and maintenance, but it is a time taking process.

This tool offers integration to other HP products like Load runner and UFT.



### Features of Quality center

Following are the commonly used features of HP quality center/ ALM:

- This tool is used to check the test configurations.
- With the help of this tool, we can manage the release and authoring the test and execution.
- This tool helps in project planning and tracking.
- It provides cross-project customization and test resources.
- It is used to manage risk-based quality and lab management.

### RTH

RTH is another web-based open-source tool, and it stands for **the Requirement and testing hub**. It is used to manage the requirements, test results, and also have bug tracking facilities. This tool follows a structured approach to extend the visibility of the testing process with the help of a common repository for test cases, test result, requirements, and test plans.



### Testpad

It is a test plan tool which helps us to identify the defects or bugs. It is a combination of simple checklists and spreadsheets. And these spreadsheets are used by the developers for making the notes. While performing exploratory testing, regression testing, and Adhoc testing, this tool is a perfect choice for the test engineer because it provides the keyboard-driven interface and checklist approach.



### Features of Testpad tool

- With the help of the Testpad tool, we can create our templates.
- This tool provides secure hosting, secure communication, and reliable data.

- Testpad is mobile and tablet friendly.
- In the test pad, we can upload the screenshots and images.
- We can easily copy and paste the data from the word, excel, or any other test files.
- It has a keyboard-driven editor having JavaScript that is the responsive User interface.
- This tool invites the guest tester if we don't need the accounts.
- This tool is organized with drag 'n' drop and group the checklists into the folder.

### TestMonitor

It is a powerful test management tool, which helps us to manage a wide range of test cases, milestone, and test runs. With the help of this tool, we can get a real-time understanding of our testing process. It is a test editor, which can run the test case within minutes. For the tester, it offers a simple interface built where the tester can execute the test case at any time, any place with no required experience.



### Features of TestMonitor tool

Some of the commonly used features of TestMonitor tool are as follows:

- This tool offers the requirement specification to execute the test case and analysis of the risks.
- This tool will monitor our planning and progress.
- It gives a complete result tracking.
- It is designed to make the testing process more effective, faster, and more structured than excel.
- This tool has a reporting capabilities.

### PractiTest

This tool is used to increase the end-to-end visibility of our complete testing process and also see the test execution in real-time. With the help of this tool, we can only focus on testing rather than reporting or releasing our product on time.



## Features of PractiTest tool

Following are some critical features of PractiTest:

- This tool is used to track the bug and issues by simply e-mailing the system, and it provides the advance filters which help us to stop the duplicate bugs even before they are reported.
- It makes sure the visibility of our project with customized dashboards and reports.
- It helps us to plan our testing process with better time management.
- This tool had a smart filter feature, which helps us to manage our work.
- PractiTest enables easy customization for our projects without writing code.

## Bug Reporting

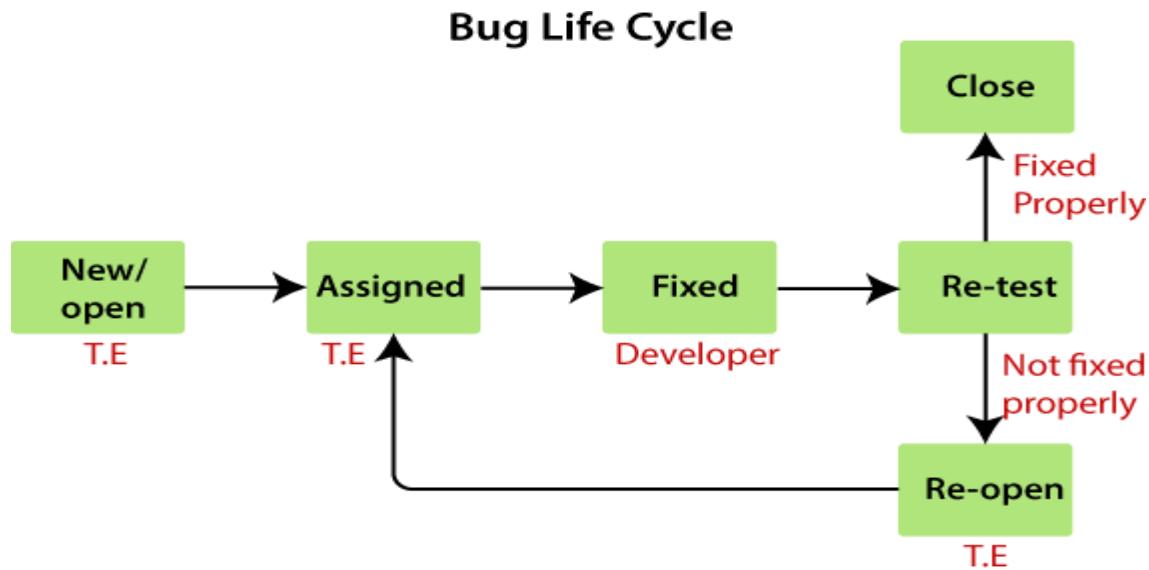
A bug report in QA is a document that describes an issue or defect in a software application. It typically includes the environment, steps to reproduce the issue, and expected versus actual behavior.

## Bug Life Cycle

bug life cycle and the different status of bugs and bug report template.

Here, we will talk about the complete life cycle of a bug from the stage it was **found, fixed, re-test, and close**.

We have some different status of bugs like **new/open**, **assigned**, **fix**, **re-open**, and **closed**.



As soon as the test engineer finds the bug, status is given as New, which indicates that a bug is just found.

This new bug needs to be reported to the concerned Developer by changing the status as **Assigned** so that the responsible person should take care of the bug.

Then the Developer first go through the bug, which means that the Developers read all the navigation steps to decide whether it is a valid bug or not.

Based on this, if the bug is valid, the Developer starts reproducing the bug on the application, once the bug is successfully reproduced, the Developer will analyze the code and does the necessary changes, and change the status as **Fixed**.

Once the code changes are done, and the bug is fixed, the test engineer re-test the bug, which means that the test engineer performs the same action once again, which is mentioned in the bug report, and changes the status accordingly:

**Close**, if the bug fixes properly, and functionally working according to the requirement.

**OR**

**Re-open**, if the bug still exists or not working properly as per the requirement, then the bug sends it back to the Developer once again.

This process is going on continuously until all the bugs are fixed and closed.

#### Whom to assign the bug

The bug can be assigned to the following:

- Developers
- Developers lead
- Test lead

**Developers:** If we know who has developed that particular module.

**Developer lead:** If we don't know the Developer who has developed the particular module.

**Test lead:** When we don't have any interaction with the development team.

When the bug is **fixed and closed** or if it is having any impact on the other module, then we go for a new bug report.

**OR**

When the status of the bug is **Re-open (not fixed)** and affecting another module, then we have to prepare the new bug report.

**Another status of the bug**

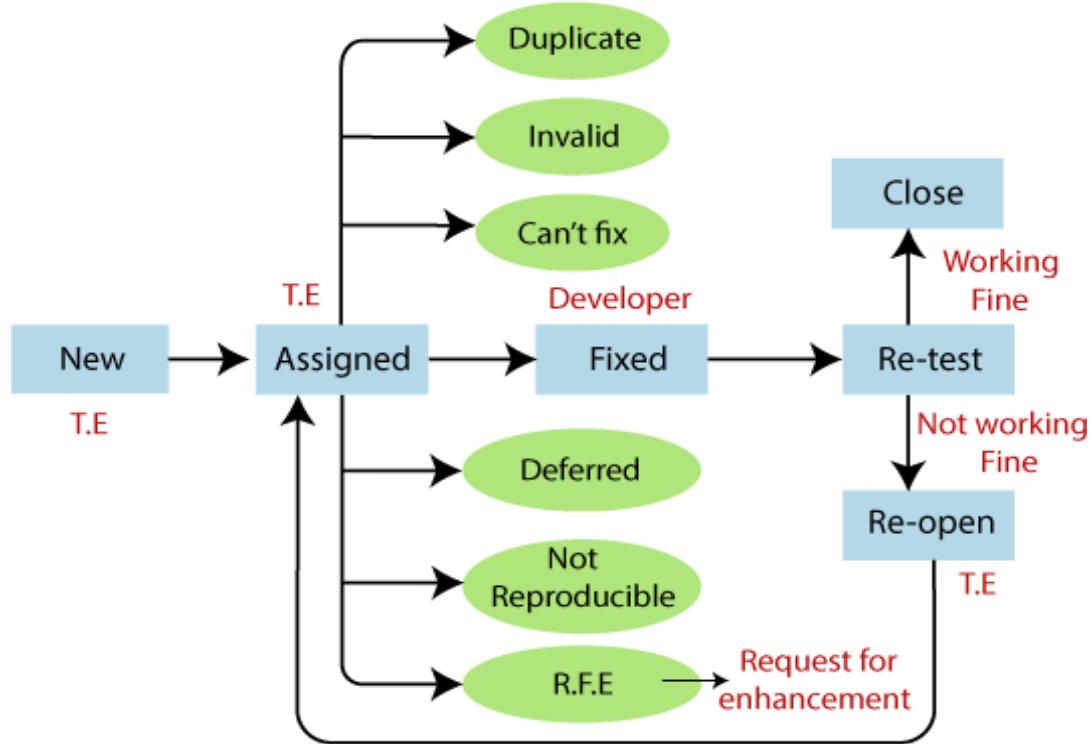
Once we prepared a bug report and send it to the Developers, the Developer will accept the bug and starts doing the necessary code changes that become the **positive flow** of the bug life cycle.

There may be several conditions where Developers may not do the necessary code changes and depend on the situation, which becomes a **negative flow or status** of the bug life cycle.

Following are the different status of the bug life cycle:

- **Invalid/rejected**
- **Duplicate**
- **Postpone/deferred**
- **Can't fix**
- **Not reproducible**
- **RFE (Request for Enhancement)**

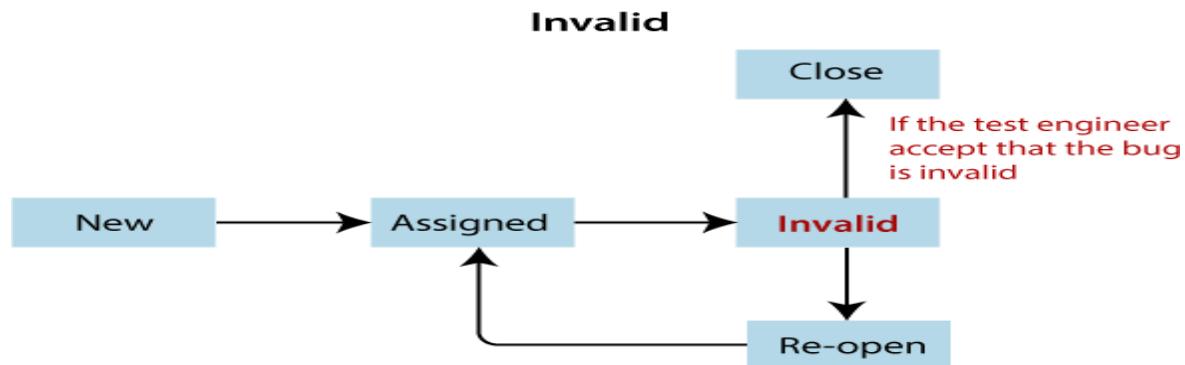
## Final Diagram of Bug life cycle



### Invalid / rejected

When the Test Engineer wrote an incorrect Bug Report because of misunderstanding the requirements, then the Developer will not accept the bug, and gave the status as **Invalid** and sent it back. (Sometime Developer can also misunderstand the requirements).

Any bug which is not accepted by the developer is known as an invalid bug.

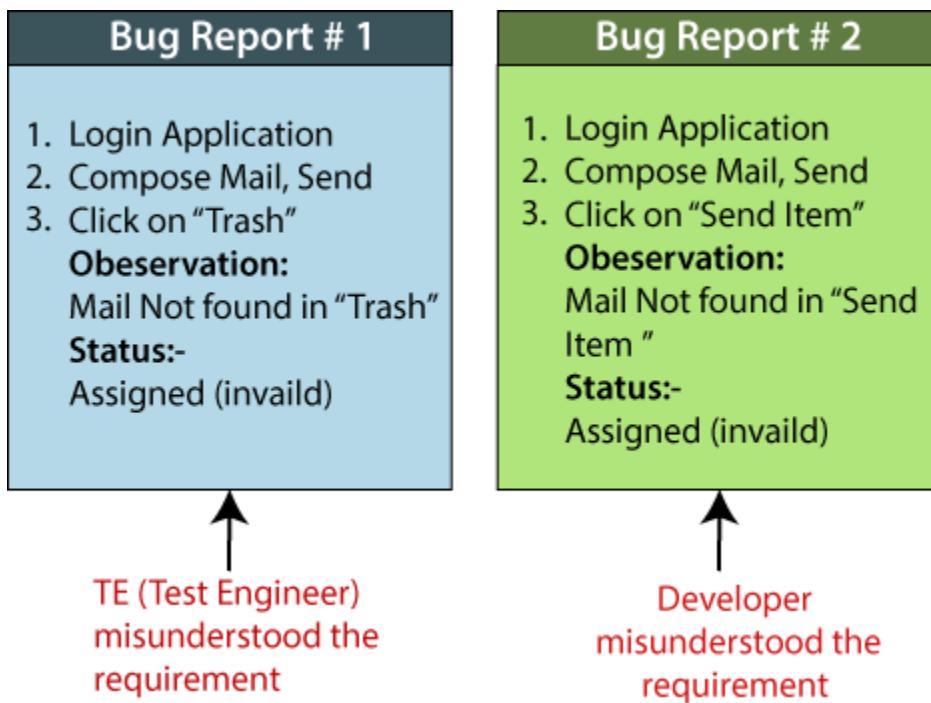


### Reasons for an invalid status of the bug

The invalid status of the bug is happened because of the following reasons:

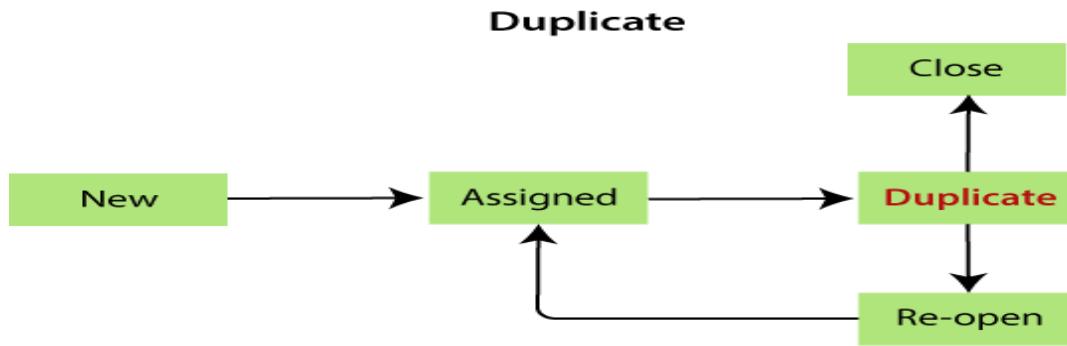
- **Test Engineer misunderstood the requirements**
- **Developer misunderstood the requirements**

Let's see one example where the test engineer and developer misunderstood the requirements as we can see in the below image:



### Duplicate

When the same bug has been reported multiple times by the different test engineers are known as a **duplicate** bug.



### Reasons for the duplicate status of the bug

Following are the reasons for the duplicate status:

- **Common features:**

**For example:** Suppose we have test engineer P and Q which are testing the software, the test engineer P and Q will test their features like **login the application**.

Here, the test engineer P enters the valid username and password, and click on the login button.

Once P click on the login button, it opens a blank page, which means that it is a bug.

After that, P prepares a bug report for the particular bug and sends it to the developer.

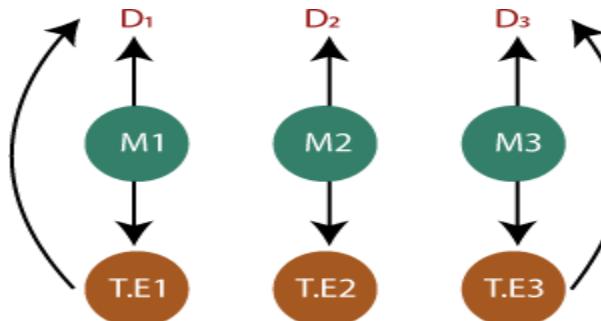
Then the test engineer Q also login the application and got the same bugs. Q also prepare a bug report and send it to the developer.

Once the developer got both test engineers bug report, he/she sends back the bug report to the Q and say it is duplicate.

- **Dependent Modules**

As we can see in the below image, that the test engineer wants to **compose a mail**, so first, the test engineer needs to **login**, then only he/she can able to compose a mail.

If the bug is found in the **login module**, the test engineer cannot do further process because the composing module is dependent on the login module.



**Compose  
Inbox → Chat  
Send Items  
Draft**

**Login  
Help  
Setting  
Logout**

○

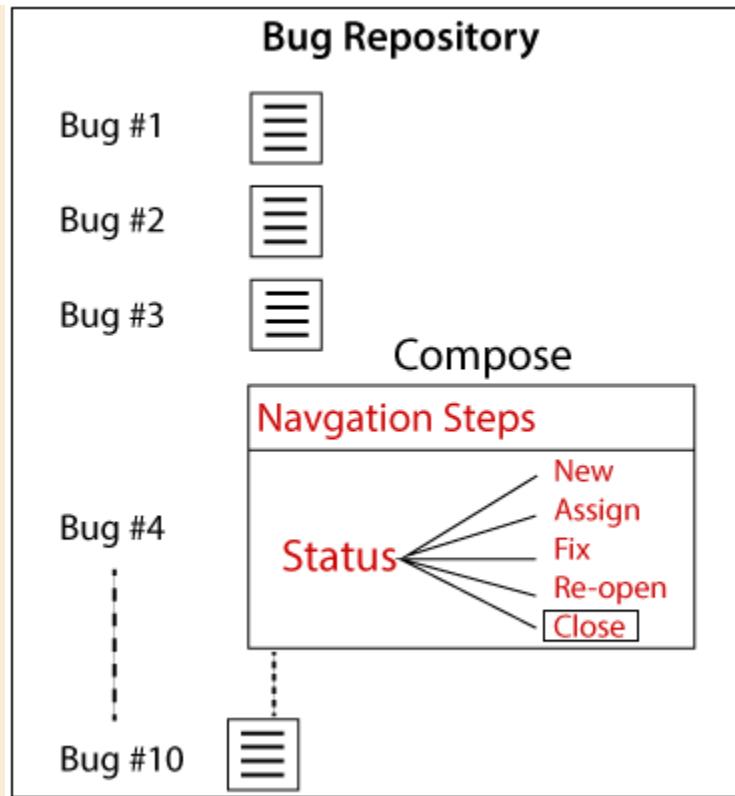
- **To avoid the duplicate bug**

If the Developer got the duplicate bug, then he/she will go to the bug repository and search for the bug and also check whether the bug exist or not.

If the same **bug exist**, then no need to log the same bug in the report again.

**Or**

If the bug **does not exist**, then log a bug and store in the bug repository, and send to Developers and Test Engineers adding them in [CC].



### Not Reproducible

The Developer accepts the bug, but not able to Reproduce due to some reasons.

These are the bug where the developer is not able to find it, after going through the navigation step given by the test engineer in the bug report.

#### Reasons for the not reproducible status of the bug

Reasons for the not reproducible status of the bug are as follows:

- **Incomplete bug report**

The Test engineer did not mention the complete navigation steps in the report.

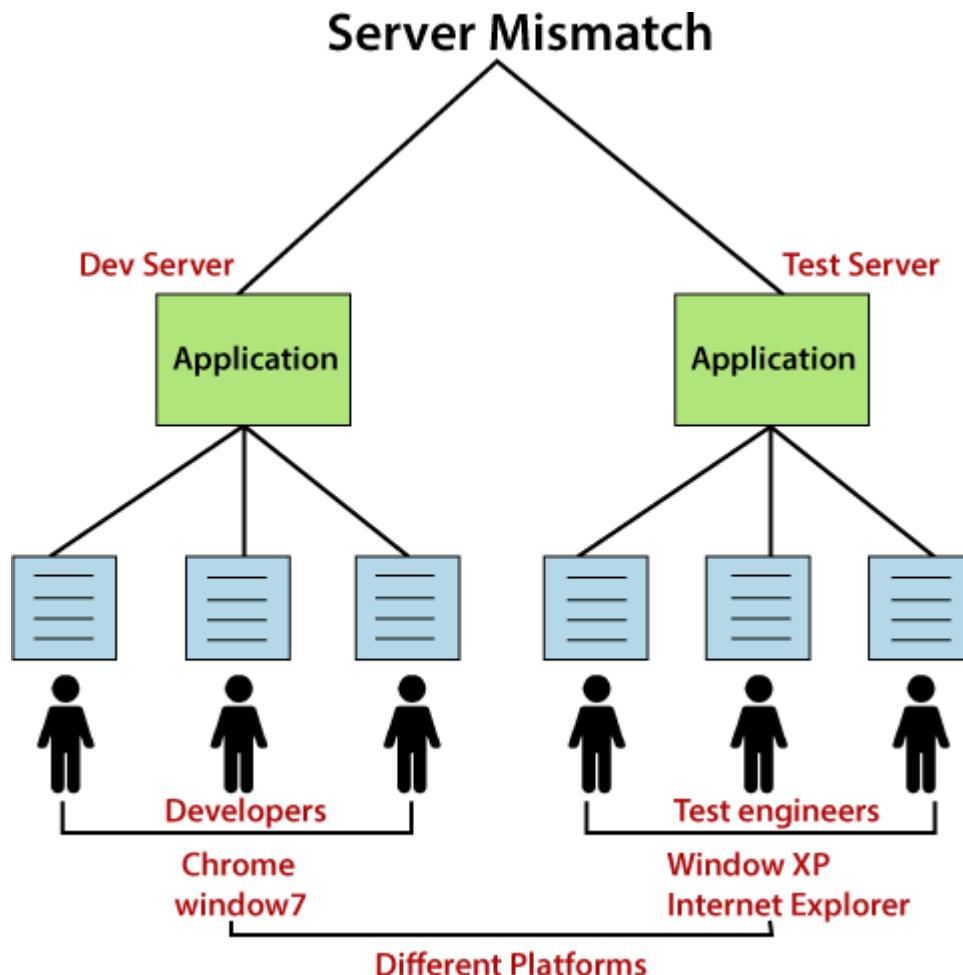
- **Environment mismatch**

Environment mismatch can be described in two ways:

- **Server mismatch**

- **Platform mismatch**

**Server mismatch:** Test Engineer is using a different server (**Test Server**), and the Developer is using the different server (**Development Server**) for reproducing the bug as we can see in the below image:



**Platform mismatch:** Test engineer using the different Platform (**window 7 and Google Chrome browser**), and the Developer using the different Platform (**window XP and internet explorer**) as well.

- **Data mismatch**

Different Values used by test engineer while testing & Developer uses different values.

**For example:**

The requirement is given for admin and user.

|                                     |                                       |
|-------------------------------------|---------------------------------------|
| Test engineer(user) using the below | the Developer (admin) using the below |
|-------------------------------------|---------------------------------------|

|                                   |                                   |
|-----------------------------------|-----------------------------------|
| requirements:                     | requirements:                     |
| User name → abc<br>Password → 123 | User name → aaa<br>Password → 111 |

I.e., both are using a different value for the same login module.

- **Build mismatch**

The test engineer will find the bug in one Build, and the Developer is reproducing the same bug in another build. The bug may be automatically fixed while fixing another bug.

- **Inconsistent bug**

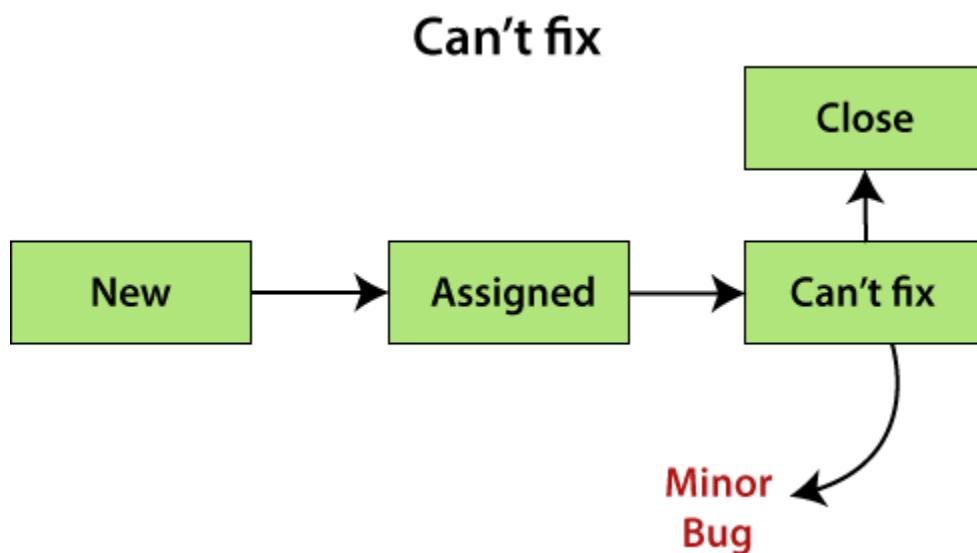
The Bug is found at some time, and sometime it won't happen.

**Solution for inconsistent bug:**

As soon as we find the bug, first, **take the screenshot**, then developer will **re-confirm** the bug and fix it if exists.

### Can't fix

When Developer accepting the bug and also able to reproduce, but can't do the necessary code changes due to some constraints.



### Reasons for the can't fix status of the bug

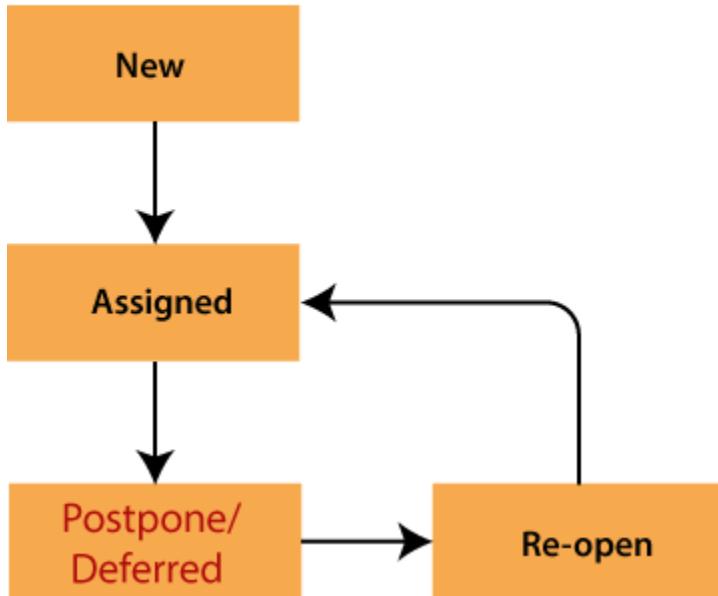
Following are the constraints or reasons for the can't fix bug:

- **No technology support:** The programming language we used itself not having the capability to solve the problem.
- **The Bug is in the core of code (framework):** If the bug is **minor** (not important and does not affect the application), the development lead says it can be fixed in the next release.  
Or if the bug is **critical** (regularly used and important for the business) and development lead cannot reject the bug.
- **The cost of fixing a bug is more than keeping it.**

### Deferred / postponed

The deferred/postpone is a status in which the bugs are postponed to the future release due to time constraints.

## Postpone/Defferred

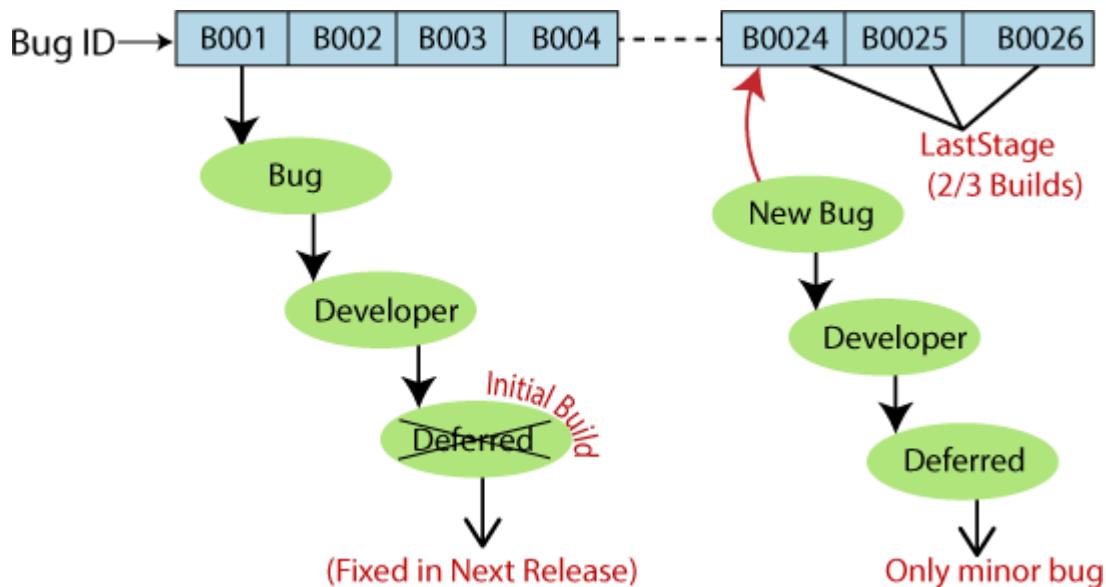


The deferred status of the bug was not fixed in the initial build because of the time constraints.

As we can see in the below image:

The **Bug ID-B001** bug is found at the initial build, but it will not be fixed in the same build, it will postpone, and **fixed in the next release**.

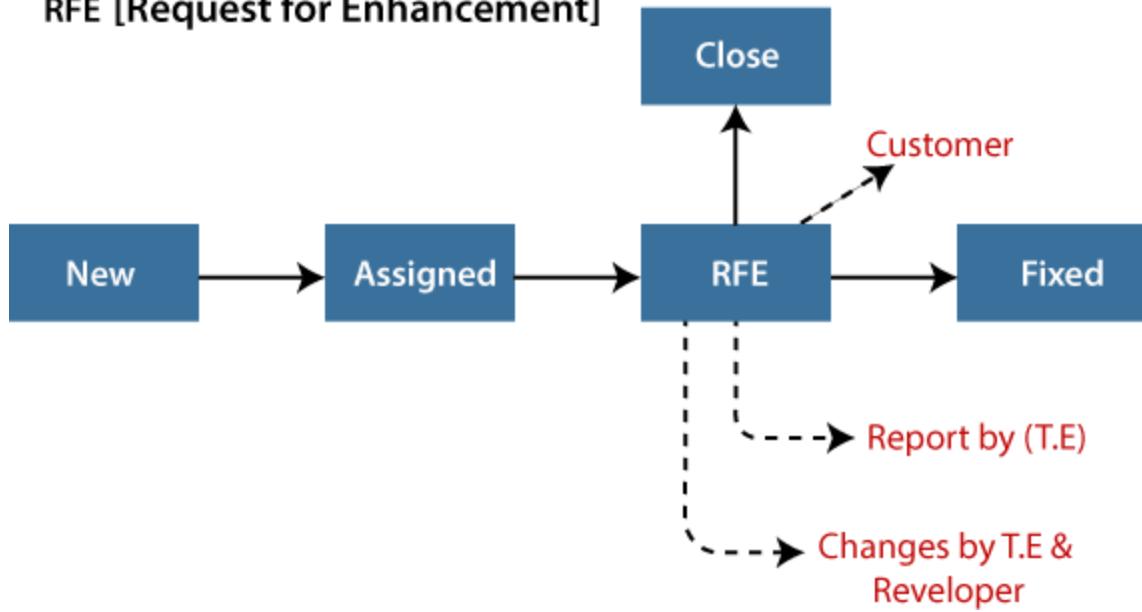
And **Bug ID- B0024, B0025, and B0026** are those bugs, which are found in the last stage of the build, and they will be fixed because these bugs are the minor bugs.



### RFE (Request for Enhancement)

These are the suggestions given by the test engineer towards the enhancement of the application in the form of a bug report. The RFE stands for **Request for Enhancement**.

## RFE [Request for Enhancement]

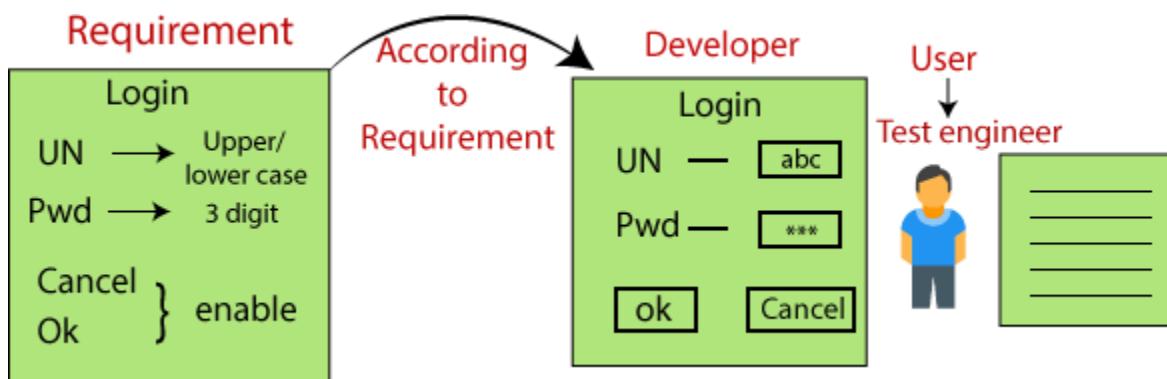


As we can see in the below example image that the test engineer thinks that the look and feel of the application or software are not good because the test engineer is testing the application as an end-user, and he/she will change the status as **RFE**.

And if the customer says **Yes**, then the status should be **Fix**.

Or

If the customer says **no**, then the status should be **Close**.



## Bug Report Template (excel)

The bug report template is as follows:

| <b>Bug Report Template</b> |  |
|----------------------------|--|
| <b>Bug ID</b>              |  |
| <b>Module</b>              |  |
| <b>Requirements</b>        |  |
| <b>Test Case Name</b>      |  |
| <b>Release</b>             |  |
| <b>Version</b>             |  |
| <b>Status</b>              |  |
| <b>Reporter</b>            |  |
| <b>Date</b>                |  |
| <b>Assign To</b>           |  |
| <b>CC</b>                  |  |
| <b>Severity</b>            |  |
| <b>priority</b>            |  |
| <b>Server</b>              |  |
| <b>Platform</b>            |  |
| <b>Build No.</b>           |  |
| <b>Test Data</b>           |  |
| <b>Attachment</b>          |  |
| <b>Brief Description</b>   |  |
| <b>Navigation Steps</b>    |  |
| <b>Observation</b>         |  |
| <b>Expected Result</b>     |  |
| <b>Actual Result</b>       |  |
| <b>Additional Comments</b> |  |

Let see one example of the bug report:

|                       |                          |
|-----------------------|--------------------------|
| <b>Bug ID</b>         | Boo12                    |
| <b>Module</b>         | Login                    |
| <b>Requirement</b>    | 1                        |
| <b>Test case name</b> | Gmail_login_compose_mail |
| <b>Reporter</b>       | Test engineer name       |
| <b>Release</b>        | delta                    |

|                          |                                                                                                                                                  |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Version</b>           | 3.0                                                                                                                                              |
| <b>Status</b>            | assigned                                                                                                                                         |
| <b>Date</b>              | 23-02-2020                                                                                                                                       |
| <b>Assign to</b>         | YY developer                                                                                                                                     |
| <b>CC</b>                | Test lead, developer                                                                                                                             |
| <b>Severity</b>          | critical                                                                                                                                         |
| <b>Priority</b>          | P2                                                                                                                                               |
| <b>Platform</b>          | Window XP, internet explorer7.0                                                                                                                  |
| <b>Build no.</b>         | B02                                                                                                                                              |
| <b>Test data</b>         | Username=xyz, password= 123                                                                                                                      |
| <b>Brief description</b> |                                                                                                                                                  |
| Navigation steps         | <ul style="list-style-type: none"> <li>○ Login Gmail application</li> <li>○ Compose mail and confirmation message should be displayed</li> </ul> |
| observation Mail         | not found in inbox                                                                                                                               |
| Expected result Mail     | should also be in the inbox.                                                                                                                     |
| Actual result            | Mail not found in inbox                                                                                                                          |

**Additional comments** -----

Here, we are describing some important attributes of the bug report.

**Bug ID:** it is a unique number given to the bug.

**Test case name:** When we find a bug, we send a bug report, not the test case to the concerned developer. It is used as a reference for the test engineer.

**Severity:** It is the impact of a bug on the application. It can be a blocker, critical, major, and minor.

**Priority:** In this, we have to decide which bug has to be fixed first. It could be P1/P2/P3/P4, urgent, high, medium, and low.

**Status:** The different status of the bug which could be assigned, invalid, duplicate, deferred, and so on.

**Reporter:** In this, we will mention the name of the person who found the bug. It could be the test engineer, and sometime it may be a developer, business analyst, customer, etc.

**Date:** It provides the date when the bug is found.

**Release/Build Version:** It provides the release number in which the bug occurs, and also the build version of the application.

**Platform:** Mention the platform details, where we exactly find the bug.

**Description:** In this, we will explain the navigation steps, expected and actual results of the particular bug.

**Attachments:** Attach the screenshots of the bug, which we captured because it helps the developers to see the bug.

### The Drawback of a manual bug report

Following are the disadvantages of manual bug report:

- **Time consuming**

While searching every bug in the bug report, it will be time taken process.

- **Possibility of human error**  
A bug may be repeated, wrong data mentioned in the bug report, and miss something to add on the bug report.
- **No security**  
Anyone can change it or delete it.
- **Tedious process**
- **No centralized repository**

## UNIT IV ADVANCED TESTING CONCEPTS

6

Performance Testing: Load Testing, Stress Testing, Volume Testing, Fail-Over Testing, Recovery Testing, Configuration Testing, Compatibility Testing, Usability Testing, Testing the Documentation, Security testing, Testing in the Agile Environment, Testing Web and Mobile Applications

### Performance Testing

It is the most important part of non-functional testing..

While doing performance testing on the application, we will concentrate on the various factors like **Response time, Load, and Stability** of the application.

**Response time:** Response time is the time taken by the server to respond to the client's request.

**Load:** Here, Load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.

**Stability:** For the stability factor, we can say that, when N-number of users using the application simultaneously for a particular time.

#### When we use performance testing?

We will do performance testing once the software is stable and moved to the production, and it may be accessed by the multiple users concurrently, due to this

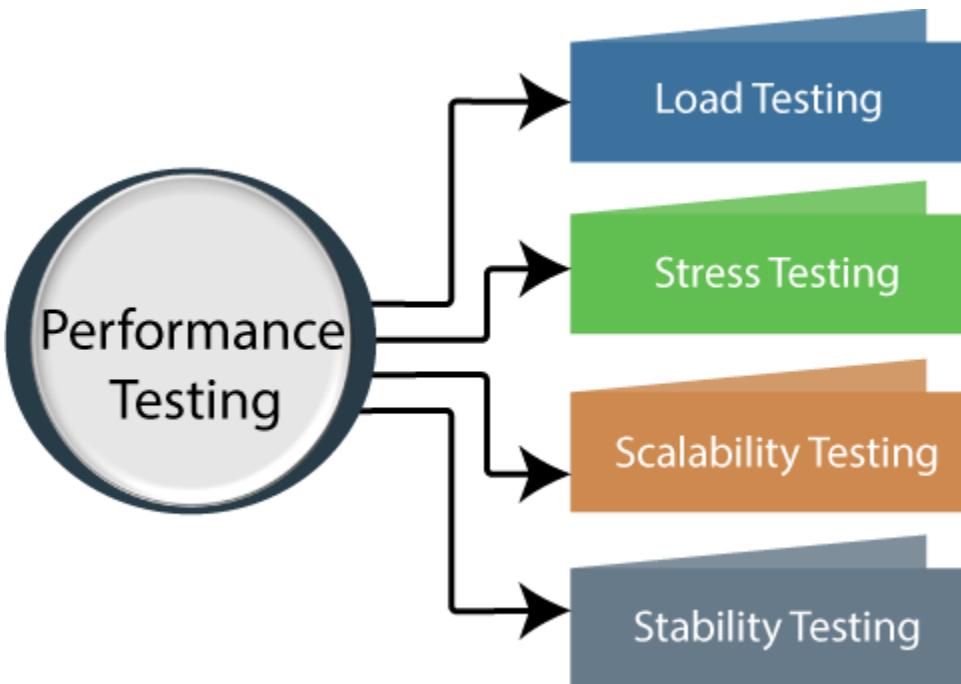
reason, some performance issues may occur. To avoid these performance issues, the tester performs one round of performance testing.

Since it is non-functional testing which doesn't mean that we always use performance testing, we only go for performance testing when the application is functionally stable.

### Types of Performance Testing

Following are the types of performance testing:

- **Load testing**
- **Stress testing**
- **Scalability testing**
- **Stability testing**

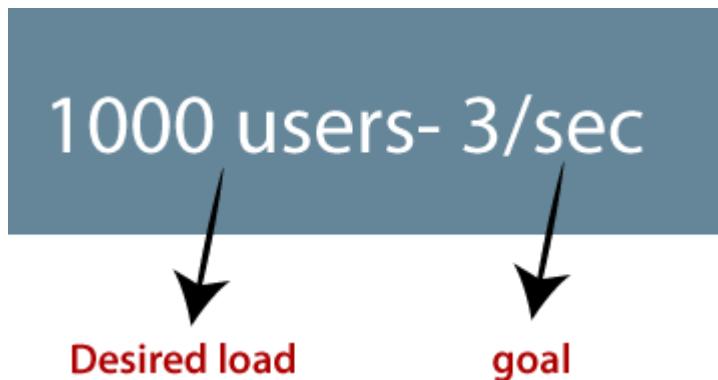


Let us discuss one by one to give you a complete understanding of **Load**, **Stress**, **Scalability**, and **Stability** performance testing.

#### Load testing

The load testing is used to check the performance of an application by applying some load which is either less than or equal to the desired load is known as load testing.

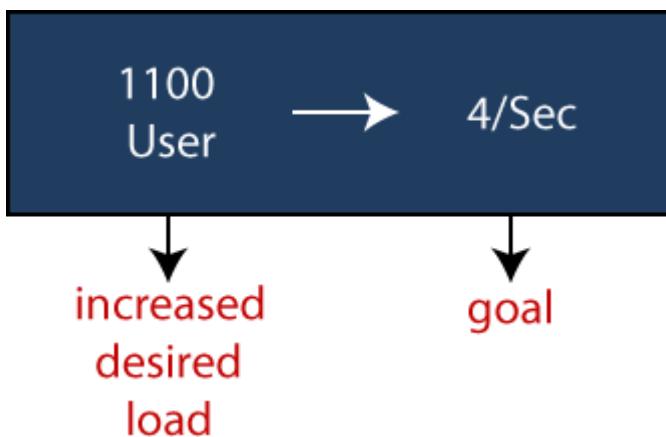
**For example:** In the below image, **1000 users** are the **desired load**, which is given by the customer, and **3/second** is the **goal** which we want to achieve while performing a load testing.



### Stress Testing

The stress testing is testing, which checks the behavior of an application by applying load greater than the desired load.

**For example:** If we took the above example and increased the desired load 1000 to 1100 users, and the goal is 4/second. While performing the stress testing in this scenario, it will pass because the load is greater (100 up) than the actual desired load.



### Scalability Testing

Checking the performance of an application by increasing or decreasing the load in particular scales (no of a user) is known as **scalability testing**. Upward scalability and downward scalability testing are called scalability testing.

Scalability testing is divided into two parts which are as follows:

- **Upward scalability testing**
- **Downward scalability testing**

### Upward scalability testing

It is testing where we **increase the number of users on a particular scale** until we get a crash point. We will use upward scalability testing to find the maximum capacity of an application.

### Downward scalability testing

The downward scalability testing is used when the load testing is not passed, then start **decreasing the no. of users in a particular interval** until the goal is achieved. So that it is easy to identify the bottleneck (bug).

### Stability Testing

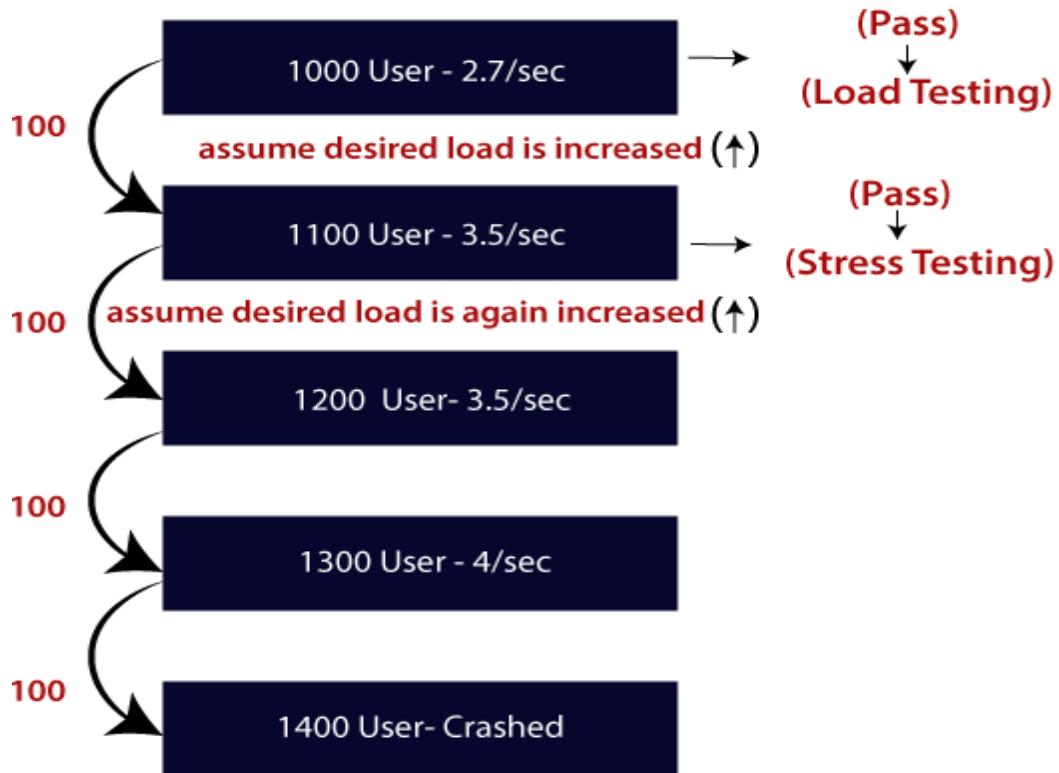
Checking the performance of an application by **applying the load for a particular duration of time** is known as **Stability Testing**.

### Performance testing example

Let us take one example where we will **test the behavior of an application where the desired load is either less than 1000 or equal to 1000 users**.

In the below image, we can see that the **100 up users** are increased continuously to check the **maximum load**, which is also called **upward scalability testing**.

- **Scenario1:** When we have the 1000 users as desired load, and the 2.7/sec is goal time, these scenarios will pass while performing the load test because in load testing, we will concentrate on the no. of users, and as per the requirement it is equal to 1000 user.
- **Scenario2:** In the next scenario, we will increase the desired load by 100 users, and goal time will go up to 3.5\sec. This scenario will pass if we perform stress testing because here, the actual load is greater than (1100) the desired load (1000).
- **Scenario3:** In this, if we increase the desired load three times as **1200 → 3.5\sec**: [it is not less than or equal to the desired load that's why it



**will Fail]**

**1300 → 4\sec:** [it is not less than or equal to the desired load. i.e., **Fail**]

**1400 → Crashed**

### Volume testing

Volume testing is testing, which helps us to check the behavior of an application by inserting a massive volume of the load in terms of data is known as volume testing, and here, we will concentrate on the number of data rates than the number of users.

### Performance testing process

The performance testing cannot be done manually since:

- We need a lot of resources, and it became a costlier approach.

- And the accuracy cannot maintain when we track response time manually.

The Performance testing process will be completed in the following steps:

- Identify performance scenarios
- Plan and design performance test script
- Configure the test environment & distribute the load
- Execute test scripts
- Result
- Analysis result
- Identify the Bottleneck
- Re-run test



If we perform a **positive flow** of the performance testing process, it could follow the below process:

### Identify performance scenarios

Firstly, we will identify the performance scenarios based on these below factors:

**Most commonly scenarios:** It means that we can find the performance scenarios based on the scenarios, which commonly used like in the **Gmail application**; we will perform **login, inbox, send items, and compose a mail and logout**.

**Most critical scenarios:** Critical scenarios mean regularly used and important for the business-like in Gmail application **login, compose, inbox, and logout**.

**Huge data transaction:** If we have huge data means that n-number of the users using the application at the same time.

Once we identify the performance scenarios, we will move to the next step.

### Plan and design performance test script

In this step, we will install the tools in the Test Engineer Machine and access the test server and then we write some script according to the test scenarios and run the tool.

Once we are done with writing the script, we will go to the next step.

### Configure the test environment & distribute the load

After writing the test scripts, we will arrange the testing environment before the execution. And also, manage the tools, other resources and distribute the load according to the "Usage Pattern" or mention the duration and stability.

### Execute test scripts

Once we are done with distributing the load, we will execute, validate, and monitor the test scripts.

### Result

After executing the test scripts, we will get the test result. And check that the result meeting the goal in the given response time or not, and the response time could be maximum, average, and minimum.

If the response is not meeting the required time response, then we will go for the **negative flow** where will perform the below steps:

## Analysis result

First, we will analyze the test result whether it meets with the response time or not.

## Identify the Bottleneck

After that, we will identify the **bottleneck (bug or performance issue)**. And the bottleneck could occur because of these aspects like the **problem in code, hardware issue (hard disk, RAM Processor), network issues**, and the **software issue (operating system)**. And after finding the bottleneck, we will perform **tuning (fix or adjustment)** to resolve this bottleneck.

## Re-run test

Once we fix the bottlenecks, re-run the test scripts and checks the result whether it meets the required goal or not.

## The problem occurs in performance testing

While performing performance testing on the application, some problems may occur, and these problems are also called the **performance issue**.

The performance issues are as follows:

- **Response time issue**
- **Scalability issue**
- **Bottleneck**
- **Speed issue**

## Response time issue

The response time means how quickly the server respond to the client's request. If the user's request does not complete in the given response time, it might have possible that the user may be lost his/her interest in the particular software or application. That's why the application or software should have a perfect response time for responding user's request quickly.

## Scalability issue

The scalability issues occur when the application cannot take the n-numbers of users and expected user requests at the same time. That's why we will do **upward scalability testing** (check the maximum capacity of the application) and **downward scalability testing** (when expected time is not matched with the actual time).

## Bottleneck

The Bottleneck is the informal name of bug, which occurs when the application is limited by a single component and creates a bad impact on the system performance.

The main causes of bottlenecks are **software issues (issue related to the operating system)**, **hardware issues (issues related to the hard disk, RAM and the processor)**, and **coding issue**, etc.

Following are the most common performance bottlenecks:

- Memory utilization
- Disk usage
- CPU utilization
- Operating System limitations
- Network utilization

## Speed issues

When we perform performance testing on the application, the application should be faster in speed to get the user's interest and attention because if the application's speed is slow, it may lose the user interest in the application.

## Performance test tools

We have various types of performance testing tools available in the market, where some are commercial tools and open-source tool.

**Commercial tools: LoadRunner[HP], WebLOAD, NeoLoad**

**Open-source tool: JMeter**

## LoadRunner

It is one of the most powerful tools of performance testing, which is used to support the performance testing for the extensive range of protocols, number of technologies, and application environments.

It quickly identifies the most common causes of performance issues. And also accurately predict the application scalability and capacity.

### JMeter

The Apache JMeter software is an open- source tool, which is an entirely a Java application designed to load the functional test behavior and measure the performance.

Generally, it was designed for testing the Web Applications but now expanded to other test functions also.

Apache JMeter is used to test performance for both static and dynamic resources and dynamic web applications.

It can be used to reproduce the heavy load on a server, network or object, group of servers to test its strength or to analyze overall performance under different load types.

### WebLOAD

WebLOAD testing tool used to test the load testing, performance testing, and stress test web applications.

The WebLOAD tool combines performance, scalability, and integrity as a single process for the verification of web and mobile applications.

### NeoLoad

Neotys develop a testing tool which is called NeoLoad. The NeoLoad is used to test the performance test scenarios. With the help of NeoLoad, we can find the bottleneck areas in the web and the mobile app development process.

The NeoLoad testing tool is faster as compared to traditional tools.

Apart from them, some other tools are **Electric load**, **web stress tool**, **LoadUI Pro**, **StresStimulus**, **LoadView**, **LoadNinja**, and **RedLine13**, which helps to test the performance of the software or an application.

## Fail-Over Testing

Failover testing is a technique that validates if a system can allocate extra resources and backup all the information and operations when a system fails abruptly due to some reason. This test determines the ability of a system to handle critical failures and handle extra servers. So, the testing is independent of the physical hardware component of a server.

It is preferred that testing should be performed by servers. **Active-active** and **active-passive** standby are the two most common configurations. Both the techniques achieve failover in a very different manner but both of them are performed to improve the server's reliability.

*For example*, if we have three servers, one of them fails due to heavy load, and then

two situations occur. Either that failed server will restart on its own or another situation when the failed server cannot be restarted, the remaining servers will handle the load. Such situations are tested during this test.

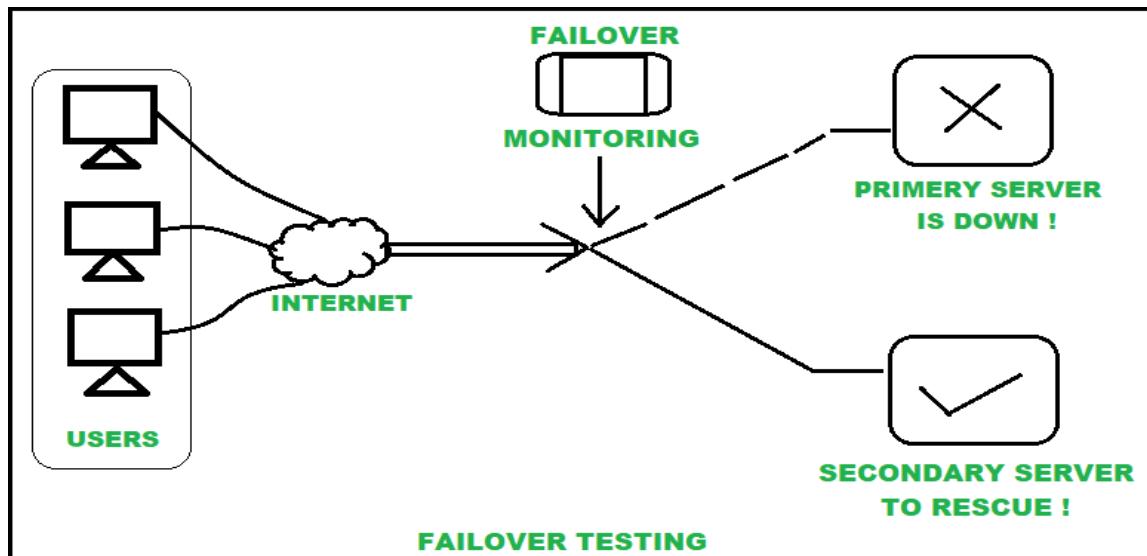
### **Considerable Factors Before Performing Failover Testing :**

1. The budget has to be the first thing to be taken into consideration before thinking about performing the Failover test.
2. The budget is connected to the frameworks that might get crashed or breakdown under pressure/load.
3. Always keep in mind that how much time it will take to fix all of the issues caused by the failure of the system.
4. Note down the most likely failures and organize the outcomes according to how much harm is caused by the failure.

### **Considerable Factors While Performing Failover Testing :**

1. Keep a plan of measures to be taken after performing a test.
2. Focus on execution of the test plan.
3. Set up a benchmark so that performance requirements can be achieved.
4. Prepare a report concerning issue requirements and/or requirements of the asset.

### **Working of Failover testing :**



1. Consider the factors before performing failover testing like budget, time, team, technology, etc.
2. Perform analysis on failover reasons and design solutions.
3. Develop test cases to test failover scenarios.
4. Based on the result execute the test plan.
5. Prepare a detailed report on failover.
6. Take necessary actions based on the report.

**Benefits of Failover Testing :**

1. Allows users to configure everything like user access and network settings and so on.
2. Ensures that the configuration made is working properly.
3. All the faults are easily resolved in the system's server beforehand.
4. Provides better services so that users' servers can run smoothly.
5. Ensures no loss during downtime.

**Examples of Failover Testing :**

1. Banking and Financial applications
2. Telecom applications
3. Visa applications
4. Trading applications
5. Emergency service business applications
6. Government applications
7. Defense service-related applications

Once the failure in a system is identified, the issue is resolved and the system gets back to its previous form. The prime purpose of this test is to ensure to retain data and resources on time when crashes happen.

## Recovery Testing

Recovery testing is testing where the test engineer will test the application to check how well the Software or the application recovers from disasters or crashes.

It is one of the most important types of **non-functional testing** categorized under **performance testing**.

In other words, we can say that the recovery testing is done to verify how fast and better the application can improve or learn the capability of the software after it has gone through any **software, hardware crashes or network failures** etc.

It is the software's required **failure** in a diversity of ways to confirm that recovery is properly performed.

While executing the recovery testing, we should first take the backup and save it to a secured location to keep away from any data loss if the data is not recovered successfully.

**The software/ hardware is forcefully failed to verify the following aspects while executing the recovery testing:**

- Lost data can be retrieved completely or not.
- It fails to verify that the fraction of scenarios in which the system can recover back.
- It is failing to verify that if any other additional operations of the software can be done or not.

### Example of Recovery testing

#### Scenario 1

Suppose we are using the browser, let say **Google Chrome**, and the power goes off. When we switch on the system again and re-open Google Chrome, we get a message window that displays whether we want to start a new session or restore the previous session.

So, in this situation, while we are restarting the system while a browser has a definite number of sessions and check if the browser can recover all of them or not.

#### **Some of the most common failures which need to test for recovery:**

Here, we list out some of the most frequent failures which needs to test, while performing the recovery testing:

- **External device not responding**
- **Network issue**
- **Wireless network signal loss**
- **Power failure**
- **Server not responding**
- **Database overload**
- **External server not reachable**
- **DLL file missing**

- **Physical conditions**
- **Stopped services**

### Why recovery testing is important?

The recovery testing is significant if we are developing the application for a user who will decides the difference between success and failure for our organization. Therefore, we need to develop software, which has enough consistency and recoverability.

### When do we need to perform the recovery testing?

- Whenever we have a release or upgrade, we need to review the system-specific recovery testing.
- To perform recovery testing, we need to ensure that everyone is included in each other's roles and responsibilities.
- Critical business functions can normally run when there is a failure or disaster.

### Who implements Recovery testing?

The Recovery testing is a part of **Business Continuity Planning (BCP)** and involves a host of roles. The following concerned persons can perform the recovery testing:

- It can be executed by the **Production Service and Service Management teams** who have the knowledge and experience in managing such outages.
- The **Technical SMEs** recognize the core systems maintained by the hardware.
- The recovery testing can perform by the **IT operations teams** which manage servers and other hardware infrastructure.

### Recovery testing life cycle

The recovery testing life cycle includes the various phase, which is as follows:

- **Standard operations**
- **Disaster and failure occurrence**
- **Interruption to Standard Process**
- **Recovery Process**

- **Rebuild Process**

Let's understand them one by one in details:



- **Standard Operations**

In the standard operations phase, we will establish the system according to the Software and hardware requirements, where the particular system can execute as expected. It is used to define the way system is planned to work.

- **Disaster and failure Occurrence**

In the next phase of recovery testing, we can identify the several failures of the system and those failures are as follows:

- **Power failure**
- **Hardware failure**
- **Physical conditions**
- **Server not reachable and many more.**

- **Interruption to Standard Process**

It leads us to losses in terms of business, financial losses, relations with the client, reputation in the market, etc.

- **Recovery Process**

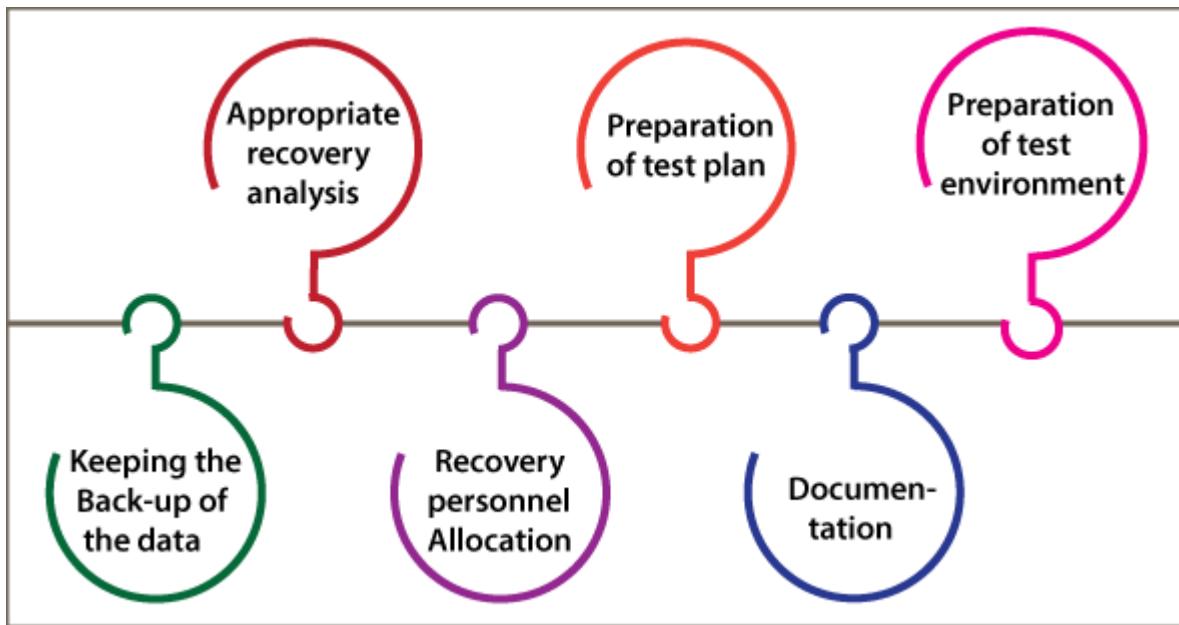
In recovery testing, the **recovery process** is used to keep away from the most important losses in companies and have backup plans with minimal impact on the interruption system.

- **Rebuild Process**

The last phase of the recovery testing process is the rebuild process, which contains the already specified documents and processes that have to be followed. And in this phase, all configuration files and folders are restored to retrieve the lost data.

#### Steps to be performed before implementing a Recovery Testing:

The following steps need to be performed before executing the recovery testing process to ensure the performance of recovery testing:



#### Step1: Appropriate recovery analysis

Before implementing the recovery testing, we should make sure that the proper analysis has to be done to verify the possibility of recovery. The recovery analysis is necessary for our better understanding of the recovery-related modification, which can impact the system's working.

To complete the appropriate analysis, we can observe the below aspects:

- The system's capability to allocate additional resources such as servers in case of critical failures or additional CPUs.
- How to track the failures.
- The effect of the failures, failures that can occur, and solutions to the failure.

## Step2: Preparation of Test Plan

In the next step, we will prepare the Test cases according to the analysis results, which we discussed in the above step.

## Step3: Preparation of Test environment

After preparing the test cases, we moved to our next step to design the test environment as per the **recovery analysis results**.

## Step4: Keeping the Back-up of the data

After the preparing the test cases and test environment, we can go to our next step to keep the back-up data related to the software, for example, **several states of the software and database**.

Similarly, if the data is significant, then **we can keep the backup data depending upon the criticality, with the help of the below strategies:**

- The backing up of the data at one or various locations.
- Single back up or Multiple back-ups.
- Automatic set up for back up at every **n** minute, for example, 20 minutes.
- Online and Offline backups.
- To perform and track the backups, we can have a separate team.
- Allocation of resources for recovery testing.

## Step5: Recovery personnel Allocation

Once the backup of data is sustained successfully, we will have enough knowledge for the recovery testing being conducted and allocate the recovery personnel.

## Step6: Documentation

In the last step, we will document all the steps performed before and throughout the recovery testing because, in case of a failure, the system can be tested for its performance.

## Advantages and Disadvantages of Recovery testing

### Advantages

Some of the major **benefits** of performing the recovery testing are as follows:

- One of the most important advantages of **recovery testing** is to recovers system quality because whenever the bugs are detected and fixed, it improves the system's quality.
- The recovery testing will help us to remove risks.
- It helps us to decreases the risk of failure of the software product in the market.
- When we implement the recovery testing, we can easily identify the performance-related issues and fix them before the software product goes live in the market.
- The system is more stable, reliable and bug-free once we performed the recovery testing.

### Disadvantages

Following are the **disadvantages** of recovery testing:

- Recovery testing is a **time-consuming process** as the test cases are random.
- It is an expensive process.
- A skilled person is essential to test the recovery testing; if the untrained test engineer implements the recovery testing, they should have all the data for testing: data and backup files.
- In recovery testing, we may not detect all the potential bugs in a few cases because sometimes the issues are unpredictable.

### Configuration Testing

**Configuration Testing** is the type of [Software Testing](#) which verifies the performance of the system under development against various combinations of software and hardware to find out the best configuration under which the system can work without any flaws or issues while matching its functional requirements.

Configuration Testing is the process of testing the system under each configuration of the supported software and hardware. Here, the different configurations of hardware and software means the multiple operating system versions, various browsers, various supported drivers, distinct memory sizes, different hard drive types, various types of CPU etc.

**Various Configurations:****• Operating System Configuration:**

Win XP, Win 7 32/64 bit, Win 8 32/64 bit, Win 10 etc.

**• Database Configuration:**

Oracle, DB2, MySql, MSSQL Server, Sybase etc.

**• Browser Configuration:**

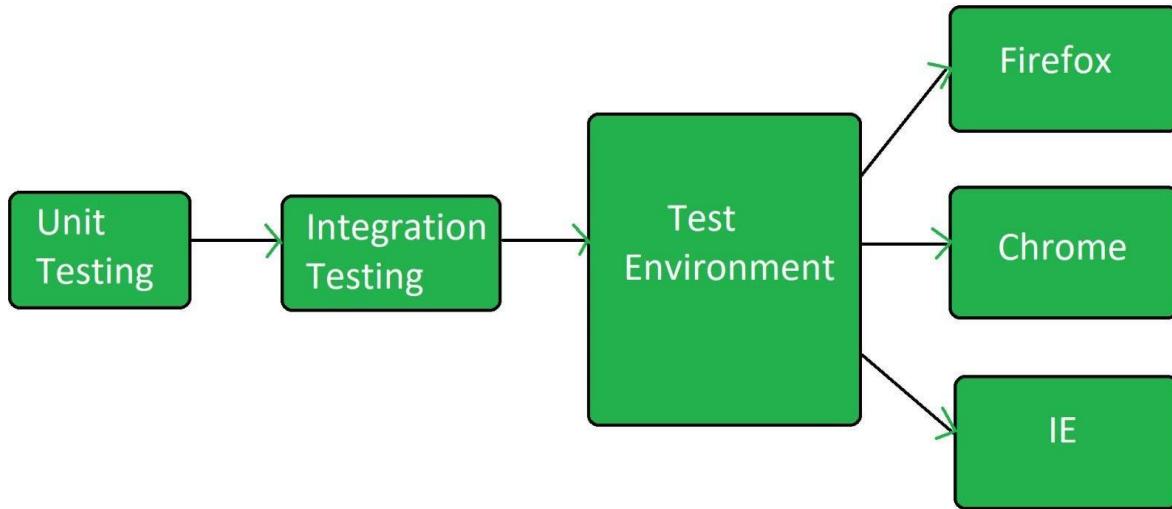
IE 8, IE 9, FF 16.0, Chrome, Microsoft Edge etc.

**Objectives of Configuration Testing:**

The objective of configuration testing is:

- To determine whether the software application fulfills the configurability requirements.
- To identify the defects that were not efficiently found during different testing processes.
- To determine an optimal configuration of the application under test.
- To do analyse of the performance of software application by changing the hardware and software resources.
- To do analyse of the system efficiency based on the prioritization.
- To verify the degree of ease to how the bugs are reproducible irrespective of the configuration changes.

**Configuration Testing Process:**



### Types of Configuration Testing:

Configuration testing is of 2 types:

#### 1. Software Configuration Testing:

Software configuration testing is done over the Application Under Test with various operating system versions and various browser versions etc. It is a time consuming testing as it takes long time to install and uninstall the various software which are to be used for testing. When the build is released, software configuration begins after passing through the unit test and integration test.

#### 2. Hardware Configuration Testing:

Hardware configuration testing is typically performed in labs where physical machines are used with various hardware connected to them.

When a build is released, the software is installed in all the physical machines to which the hardware is attached and the test is carried out on each and every machine to confirm that the application is working fine. While doing hardware configuration test, the kind of hardware to be tested is spelled out and there are several computer hardware and peripherals which make it next to impossible to execute all the tests.

Configuration Testing can also be classified into following 2 types:

**1. Client level Testing:**

Client level testing is associated with the usability and functionality testing. This testing is done from the point of view of its direct interest of the users.

**2. Server level Testing:**

Server level testing is carried out to determine the communication between the software and the external environment when it is planned to be integrated after the release.

## Usability Testing

Usability Testing is a significant **type of software testing** technique, which is comes under the **non-functional testing**.

It is primarily used in user-centered interaction design on order to check the usability or ease of using a software product. The implementation of usability testing requires an understanding of the application, as it is extensive testing.

Generally, usability testing is performed from an end-user viewpoint to verify if the system is efficiently working or not.

Checking the user-friendliness, efficiency, and accuracy of the application is known as Usability Testing.

The primary purpose of executing the usability testing is to check that the application should be easy to use for the end-user who is meant to use it, whereas sustaining the client's specified functional and business requirements.

When we use usability testing, it makes sure that the developed software is straightforward while using the system without facing any problem and makes end-user life easier.

In other words, we can say that Usability testing is one of the distinct testing techniques that identify the defect in the end-user communication of software product. And that's why it is also known as **User Experience (UX) Testing**.

It helps us to fix several usability problems in a specific website or application, even making sure its excellence and functionality.

The execution of usability testing certifies all the necessary features of a product, from testing the effortlessness of navigating a website and to validate its flow and the content in order to propose the best user experience.

Typically, the usability testing is executed by real-life users, not by the development team, as we are already aware of that the development team is the one who has created the product. Consequently, they fail to identify the more minor defects or bugs related to the user experience.

In **Usability Testing**, the **user-friendliness** can be described with the help of the following characteristics:

- **Easy to understand**
- **Easy to access**
- **Look and feel**
- **Faster to Access**
- **Effective Navigation**
- **Good Error Handling**



Let us see them one by one for our better understanding:

### **Easy to understand**

- All the features of software or applications must be visible to the end-users.

### **Easy to Access**

- A user-friendly application should be accessible by everyone.

### **Easy to Access**

- The look and feel of the application should be excellent and attractive to get the user's interest.
- The GUI of the software should be good because if the GUI is not well, the user may be lost his/her interest while using the application or the software.
- The quality of the product is up to the mark as given by the client.

### **Faster to Access**

- The software should be faster while accessing, which means that the application's response time is quick.
- If the response time is slow, it might happen that the user got irritated. We have to ensure that our application will be loaded within 3 to 6 seconds of the response time.

### **Effective Navigation**

- Effective navigation is the most significant aspect of the software. Some of the following aspects for effective navigation:
- Good Internal Linking
- Informative header and footer
- Good search feature

### **Good Error Handling**

- Handling error at a coding level makes sure that the software or the application is bug-free and robust.
- By showing the correct error message will help to enhance the user experience and usability of the application.

### **Why do we need to perform Usability Testing?**

We need usability testing because usability testing is to build a system with great user experience. Usability is not only used for software development or website development, but it is also used for product designing.

And Customers must be comfortable with your application with the following parameters.

- The flow of an Application should be good
- Navigation steps should be clear
- Content should be simple
- The layout should be clear
- Response time

And we can also test the different features in usability testing given as follows:

- How easy it is using the application
- How easy to learn application

### Features of Usability Testing

The implementation of usability testing helps us to increase the end-user experience of the particular application and software. With the help of usability testing, the software development team can quickly detect several usability errors in the system and fix them quickly.

Some other vital features of usability testing are as follows:

- It is an essential type of **non-functional testing** technique, which comes under the **black-box testing** technique in **software testing**.
- Usability testing is performed throughout the **system** and **acceptance testing** levels.
- Generally, usability testing is implemented in the early stage of the **Software Development Life Cycle (SDLC)**.
- The execution of usability testing offers more visibility on the prospects of the end-users.
- The usability testing makes sure that the software product meets its planned purpose.

- It also helps us to find many usability errors in the specified software product.
- Usability testing mainly tests the user-friendliness, usefulness, traceability, usability, and desirability of the final product.
- It offers direct input on how real-users use the software/application.
- The usability testing includes the systematic executions of the product's usability under a measured environment.

### Parameters covered by Usability Testing

In order to test the **quality, usability, user-friendliness**, and other significant factors of the software, usability testing plays an important role. And it also helps us in order to supports the organizations for delivering a more extensive services to their target audience.

However, the impact of usability testing is inadequate to these aspects, and also covered the following various constraints or parameters that help us enhance the software's productivity.

### Parameters covered by Usability Testing



1. **Efficiency**
2. **Memorability**
3. **Accuracy**
4. **Learnability**
5. **Satisfaction**

## 6. Errors

Let's see them separately in order to enhance our knowledge of usability testing:

### 1. Efficiency

The first constraint covered by the execution of usability testing is **Efficiency**. Here, the efficiency parameter explains the end-user who is an expert and can take the minimum amount of time to execute his/her fundamental or, we can say, undeveloped task.

### 2. Memorability

The second constraint that is covered by the implementation of usability testing is **Memorability**. The Memorability of an application can be beneficial or not beneficial. But, the question arises, how can we have decided the Memorability of an application is good or bad?

The below points will give the perfect answer to the above arise the question:

- When we are not asking for an application for some time and returning to the application or trying to do the simple task without any help, we can say that the **Memorability of an application is beneficial**.
- Or, if we cannot execute a simple task without any help after a duration, we can say that the **Memorability of an application is not beneficial**.

### 3. Accuracy

The next parameter covered by performing the Usability testing is **Accuracy**. The usability testing ensures that no inappropriate/irrelevant data or information exists in the product. And also, able to discover the broken links in the particular product that helps us develop the accuracy of the final product.

### 4. Learnability

Another constraint that is encompassed by usability testing is **Learnability**. In this constraint, the end-user takes a minimum amount of time to learn the fundamental task.

### 5. Satisfaction

The execution of usability testing ensures the **customer's satisfaction** as we know that the satisfied customer can easily or freely use the application.

## 6. Errors

The last and most important parameter covered by the usability testing is **Errors detection**. At this point, we try to help the end-users fix those errors they made earlier and accomplish their tasks all over again.

### Various Strategies of Usability Testing/Usability Testing Methods

Like others type of software testing contains several approaches, usability testing also involved various strategies or methods. Some of the most frequently used usability testing methods are as follows:



- **A/B Testing**
- **Hallway Testing**
- **Laboratory Usability Testing**
- **Expert Review**
- **Automated Expert Review**
- **Synchronous Remote Usability Testing**
- **Asynchronous Remote Usability Testing**

Let us summarise them one by one for our better understanding:

#### 1. A/B Testing

The first usability testing approach is **A/B Testing**, which includes creating a similar image of the product without an essential aspect from the original, which can directly affect the user performance.

A comparative analysis understands the A/B testing, and we can go through with some of the other elements such as **colour, text, or difference of the interface**.

## 2. Hallway Testing

The next method of usability testing is **Hallway Testing**. It is one of the most successful and cost-saving approaches compared to the other usability testing methods.

In hallway testing, some random people test the application without having any earlier knowledge of the product instead of skilled professionals. As a result, we will get more precise outcomes and reliable responses for further enhancement, if any of those random people test the application more efficiently.

The primary purpose behind hallway testing is to find the most crucial environments for the bugs because those bugs can make the simple features unproductive and lethargic to work with.

## 3. Laboratory Usability Testing

The third strategy of usability testing is **Laboratory Usability Testing**. The Laboratory usability testing is performed in the existence of the viewers. Generally, it is implemented by the team in an individual lab room.

In this method, the viewers are concerned about checking the performance of the test engineers regularly and reporting the results of testing to the related team.

## 4. Expert Review

Another general approach to usability testing is **Expert Review**. The Expert Review method includes the benefits of a professionals teams who have in-depth knowledge or experience in the specified field of performing usability tests.

Usability testing is consistent as the professional's knowledge is worth the expenditure when the product has a crucial feature. The organization needs to find out the user's response before releasing the product.

The specialist in a specified field is requested to test the product, give the response, and then submit the outcomes. To submit the outcomes, the expert review can also be performed remotely.

The expert review of usability testing is implemented rapidly and takes less time than the other type of usability testing because the professionals can easily identify the loopholes and discover the flaws in the product.

And that makes the particular process costly because the company needs to appoint a skilled person. So, sometimes the clients avoid this option.

### 5. Automated Expert Review

The next essential approach of **Usability Testing** is **Automated Expert Review**. As the name recommends, automated **expert review** is executed by writing automation scripts.

To execute this usability testing approach, an organization needs to appoint a resource who is well aware of writing automation scripts and developing an automation framework.

The automation test engineers write the test scripts, and when the scripts are triggered, we can easily implement the test cases. After the implementation of the test, the results are recorded and submitted.

The automated expert review is one of the successful types of usability testing because there is less human involvement, automated scripts, and fewer chances of missing any issues.

In simple words, we can specify that it is just a program-based review of all the usability constraints. However, the problem of this method is the absence of insightful reviewing when executed by persons, which makes it a slower method of testing.

It is a primarily used method of usability testing as it is not that costly compared to the **Expert Review**.

### 6. Remote Usability Testing

The next method of usability testing is **Remote Usability testing**. As the name indicates, remote usability testing takes place by people located at remote locations, which means those situated in various states or sometimes in some other countries to achieve their testing objectives.

The remote usability testing is executed remotely and also able to report the issues if identify any. In this approach, the response can be documented and submitted by random people, not by the skilled ones.

From time to time, remote testing is implemented using video conferencing. And this approach is less expensive in comparison with other types of usability testing approaches.

The remote usability testing can be divided into the following two parts, which are as discussed below:

- **Synchronous Remote Usability Testing**
- **Asynchronous remote Usability Testing**

### **Synchronous Remote Usability Testing**

The first part of remote usability testing **Synchronous Remote usability testing**. After comprehensive research on the issues related to execute the usability testing through distant locations, the synchronous remote usability testing approach was put forward.

We can use the **WebEx** tools for the video conferencing of remote web sharing. However, It needs the effectiveness of a real presence to make this collective testing process a success.

### **Asynchronous remote Usability Testing**

The second method of the Remote usability testing approach is **Asynchronous remote usability testing**.

Asynchronous remote usability testing approach help us to easily divide the user response into various demographic and performance types.

It is the most frequent method, which uses user logs, response for user interface, and testing in the user environment itself.

In maximum situations, usability testing resolves many bugs closely related to the output of performance testing processes.

### **Usability Testing Process**

The Process of usability testing is completed into few significant steps. This process will assist us in providing and creating different results for all the issues identified during the execution of testing.

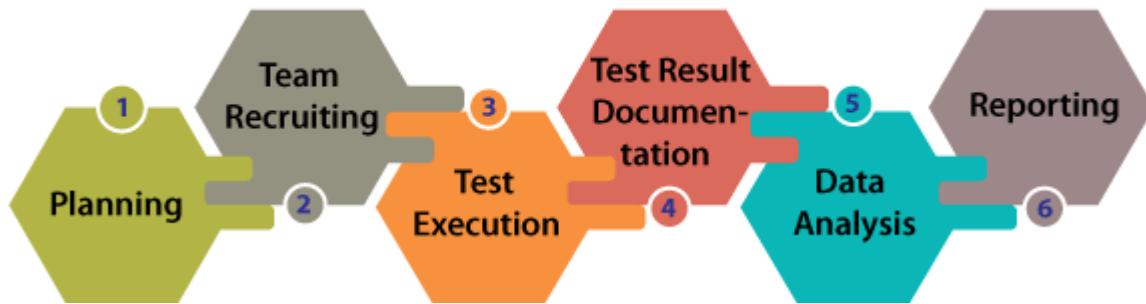
In real-time, usability testing tests the application's behavior from the user's perspective even though it is a time-consuming process, providing the tester the most precise outcomes from actual testing.

And that gives us an idea of the errors/flaws in our product and helps us distinctly before installing it on the server.

The usability testing process follows a precise set of steps to help the team get the detailed and helpful response from the end-users.

Therefore, the process of usability testing completed into the following steps, as we can see in the following image:

## Usability Testing Process



### Step1: Planning

The first step of usability testing is **Planning**, where the team makes the test plan and generates some document samples that help the testing team complete the usability testing tasks. It is one of the most essential and crucial stages in the usability testing process.

The objective of the usability test is governed in the planning step. Here, the aim is not to have the volunteers sit in front of our application and recording their activities, but we need to fix the crucial features and elements of the system.

We need to give the tasks to our test engineers who are familiar with these crucial features. And the usability testing method, number, and demographics of usability test engineers, test report formats are also fixed throughout the planning phase.

### Step2: Team Recruiting

Once the planning phase is completed, we will move to the next step of usability testing, which is **team recruiting**.

As the name suggests itself, here, we will hire or recruit the end-user delegates and the participants or the test engineers as per the budget and density of the product.

These representatives or test engineers are prepared to sit across the test sessions and validate the correctness and usability of the product.

Primarily, the selection of these test engineers is based on the necessity of testing along with the number of persons mentioned in the test plan.

As soon as the hiring of a test engineer is achieved, the team is appointed to the particular responsibilities and jobs.

### Step3: Test Execution

Once the planning and team recruiting steps have been completed successfully. We will proceed to the next step, which is **Test Execution**.

In the test execution step, the test engineers execute the usability testing and implement their assigned responsibilities. In this situation, the users' needs to test the product to find irregularity, if any, and also record them correctly.

### Step4: Test Result Documentation

The **test result documentation** step includes the results based on the **test execution** step and then proceeds for further analysis.

### Step5: Data Analysis

Once the test result documentation is done, we will move to the next step of the usability testing process, i.e., **Data Analysis**.

The response or the feedback is obtained from usability testing evaluation in the data analysis phase. And the outcomes are classified, and the patterns are acknowledged.

In this step, the data from usability tests are wholly evaluated to get expressive implications and help us provide actionable suggestions to enhance the overall usability of our product.

### Step6: Reporting

After performing all the above steps successfully, we will finally reach the last step of the usability testing process which is named as **Reporting**.

In this, we can report and share the outcomes and suggested modifications with the development team, designer, and another participant of the particular project and all the related documents along with audio, databases, screen recording, etc.

### Examples of Usability Testing

Let us see some examples, where we understand the use of usability testing.

#### Example 1

Suppose we have two applications, namely **P and Q**, that are different but perform the same job, and we will see which one is user-friendly.

Below are some of the significant parameters or constraints we look into for testing, and most of the parameters are not measurable.

- **Look & feel**
- **Navigation should be simple**
- **Speed**
- **Compatibility**
- **Help**
- **Location of components**
- **Features**

In this example, we learn the **Application P** in **4 hours**, but we take 6 hours in order to understand the **application Q**.

Let us see other different situations here to get more clarity of the above example:

- Since we understand the **application P** in 4 hours, it becomes user-friendly if we compare it to the **application Q**.
- Suppose **look and feel** is not suitable for **application P**. In such scenarios yet, we understand **application P** in 4 hours; we cannot state that application P is user-friendly.
- Thus, we look into various parameters before we say user-friendliness of a software.

In usability testing, the term **look and feel** are most commonly used term. The look & feel is used to describe that the application should be pleasant looking.

Let say we have **blue color text in the red color background**; indeed, we don't feel like using it and make a feel to the end-user to use it.

### Example 2

We are taking one **banking application** where we produce the application for the manager.

Now, if the end-user (manager) starts using the application in front of the test engineers

Suppose two test engineer sits at the back of the end-user while he/she is using the application and takes the report of the defect as a developer to check whether the end-user is using the application in a right way or not.

And the end-user (manager) will check the application step by step because he/she knows that the Test engineer is watching him/her.

Sometimes Test engineer has to do usability testing for the following reasons:

- There is no money to spend on usability testing.
- Do not want to outsource to another company.

### Example 3

In this example, the Director of the company goes and collects the software (suppose a gaming software) and distributes it to the various end-users like employees, friends, etc.

Now, these end-users will use particular game software and give their feedback to the Director.

This Director looks into their feedback, and see the major feedback, then consolidates all the feedback and makes one report.

If a particular feature for all the end-users has been reported, then that should be considered, or if the feature has been reported only by 1 or 2 end-user, then it becomes minor.

Once the consolidation of the major and minor bugs is done, they will be fixed based on the requirement of the director.

If it is a major bug, then it will fix first, or if it is minor, then it could be delayed or fixed in the next release.

### Usability Testing Checklist

The usability testing checklist contains all the documents which are related to usability testing. We don't write test cases in usability testing as we use the standard Usability testing checklist, and we are just testing the look and feel of the application.

To make the usability testing more successful, we will prepare the standard checklist, which means that "**what are the points to check**". Or if we do not make a checklist, we may miss some features in the application.

- **Create a checklist**

- **Review checklist**
- **Execute a checklist / Approve checklist**
- **Derive checklist report (Execution Report)**

Let see **one example** where we are creating a checklist for an application:

If we take the one **E-commerce application** and prepare the checklist, it would be like as below:

- All the images should have the alt tag (Tooltip).
- The Login feature should have the Forget password link.
- All the pages should have a link to the homepage of the application.
- Should be able to access all the components.

Like this, we can drive as many checklists as possible based on the product or the application.

### Bugs in Usability Testing

A usual error in usability testing is organizing a study too late in the design process. If we wait until our product is released, we won't have the time or money to resolve any issues. And we have wasted a lot of effort creating our product the wrong way.

Furthermore, we may encounter some more bugs while testing any software or application. And those bugs could be the **Path holes** and **Potential bugs**.

**Path holes and Potential bugs:** Path holes and potential bugs are those, which are visible to the developers and the test engineers while performing usability testing.

### The Advantages of Usability Testing

Some of the significant benefits of using the usability testing are as discussed below:

- The execution of usability testing helps us to validate the usability of the software.
- Its enhanced user satisfaction with the software product and also ensured to deliver a good quality product.
- The implementation of usability testing will improve the adequacy and consistency of the software product.

- With the help of usability testing, we can discover the usability issues before delivering the final product.
- The end-user is always eager to use the application.
- The execution of usability testing helps us to identify the possible bugs and defects in the software.
- It helps us to make the software more efficient and applicable.
- The usage of usability testing will help us in receiving related and precise user responses.
- It improves the adequacy and consistency of the software product.

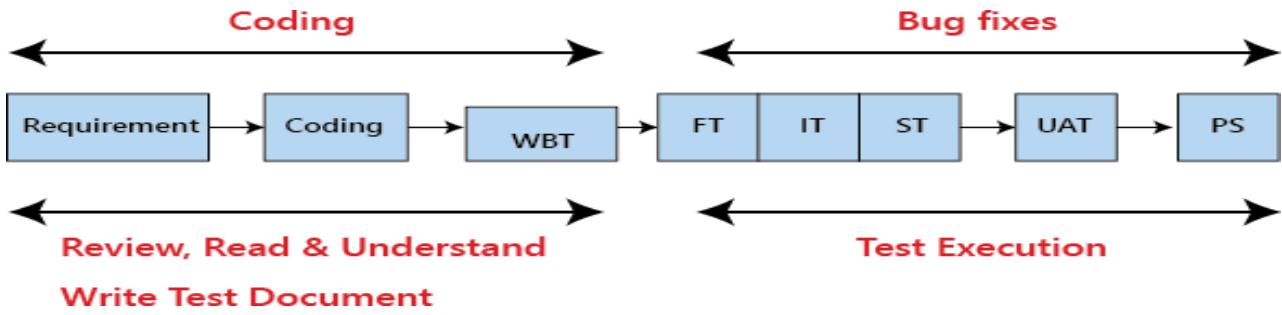
### The Disadvantage of Usability Testing

Some of most common drawbacks of implementing usability testing are as discussed below:

- As we know that budgeting is the most crucial factor while performing any software testing. Here, usability testing costing also plays an essential role. It requires many resources to establish a usability test lab, and sometimes hiring or recruiting a usability test engineer could be costly.
- As we understood from the above discussion of the usability testing that it is implemented by the end-users, sometimes it is a bit harder to identify the volunteers who can work as test engineers.
- Primarily, usability testing is not 100% representative of the actual condition.

### Testing the Documentation

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization



There is the necessary reference document, which is prepared by every test engineer before starting the test execution process. Generally, we write the test document whenever the developers are busy in writing the code.

Once the test document is ready, the entire test execution process depends on the test document. The primary objective for writing a test document is to decrease or eliminate the doubts related to the testing activities.

### Types of test document

In software testing, we have various types of test document, which are as follows:

- Test scenarios
- Test case
- Test plan
- Requirement traceability matrix(RTM)
- Test strategy
- Test data
- Bug report
- Test execution report



### Test Scenarios

It is a document that defines the multiple ways or combinations of testing the application. Generally, it is prepared to understand the flow of an application. It does not consist of any inputs and navigation steps.

For more information about test scenario, refers to the below link:

### Test case

It is an in-details document that describes step by step procedure to test an application. It consists of the complete navigation steps and inputs and all the scenarios that need to be tested for the application. We will write the test case to maintain the consistency, or every tester will follow the same approach for organizing the test document.

### Test plan

It is a document that is prepared by the managers or test lead. It consists of all information about the testing activities. The test plan consists of multiple components such as **Objectives**, **Scope**, **Approach**, **Test Environments**, **Test methodology**, **Template**, **Role & Responsibility**, **Effort estimation**, **Entry and Exit criteria**, **Schedule**, **Tools**, **Defect tracking**, **Test Deliverable**, **Assumption**, **Risk**, and **Mitigation Plan or Contingency Plan**.

For more information about the test plan, refers to the below link:

## Requirement Traceability Matrix (RTM)

The Requirement traceability matrix [RTM] is a document which ensures that all the test case has been covered. This document is created before the test execution process to verify that we did not miss writing any test case for the particular requirement.

For more information about RTM, refers to the below link:

### Test strategy

The test strategy is a high-level document, which is used to verify the test types (levels) to be executed for the product and also describe that what kind of technique has to be used and which module is going to be tested. The Project Manager can approve it. It includes the multiple components such as documentation formats, objective, test processes, scope, and customer communication strategy, etc. we cannot modify the test strategy.

### Test data

It is data that occurs before the test is executed. It is mainly used when we are implementing the test case. Mostly, we will have the test data in the Excel sheet format and entered manually while performing the test case.

The test data can be used to check the expected result, which means that when the test data is entered, the expected outcome will meet the actual result and also check the application performance by entering the in-correct input data.

### Bug report

The bug report is a document where we maintain a summary of all the bugs which occurred during the testing process. This is a crucial document for both the developers and test engineers because, with the help of bug reports, they can easily track the defects, report the bug, change the status of bugs which are fixed successfully, and also avoid their repetition in further process.

### Test execution report

It is the document prepared by test leads after the entire testing execution process is completed. The test summary report defines the constancy of the product, and it contains information like the modules, the number of written test cases, executed, pass, fail, and their percentage. And each module has a separate spreadsheet of their respective module.

### Why documentation is needed

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

### Example

Let's take a real-time example of Microsoft, Microsoft launch every product with proper user guidelines and documents, which are very explanatory, logically consistent and easy to understand for any user. These are all the reasons behind their successful products.

### Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.
- If we write the test document, we can't forget the values which we put in the first phase.
- It is also a time-saving process because we can easily refer to the text document.
- It is also consistent because we will test on the same value.

### The drawback of the test document

- It is a bit tedious because we have to maintain the modification provided by the customer and parallel change in the document.
- If the test documentation is not proper, it will replicate the quality of the application.
- Sometimes it is written by that person who does not have the product knowledge.
- Sometimes the cost of the document will be exceeding its value.

## Security testing

Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application and also help us to stop the nasty attack from the outsiders and make sure the security of our software applications.

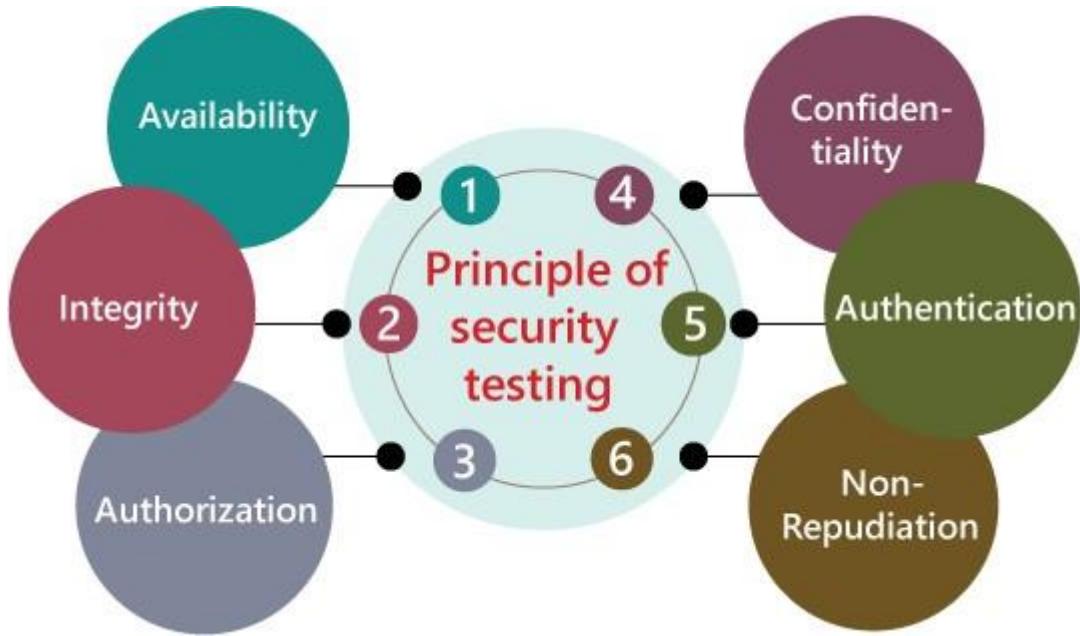
The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.

It is a testing procedure, which is used to define that the data will be safe and also continue the working process of the software.

### Principle of Security testing

Here, we will discuss the following aspects of security testing:

- Availability
- Integrity
- Authorization
- Confidentiality
- Authentication
- Non-repudiation



### Availability

In this, the data must be retained by an official person, and they also guarantee that the data and statement services will be ready to use whenever we need it.

### Integrity

In this, we will secure those data which have been changed by the unofficial person. The primary objective of integrity is to permit the receiver to control the data that is given by the system.

The integrity systems regularly use some of the similar fundamental approaches as confidentiality structures. Still, they generally include the data for the communication to create the source of an algorithmic check rather than encrypting all of the communication. And also verify that correct data is conveyed from one application to another.

### Authorization

It is the process of defining that a client is permitted to perform an action and also receive the services. The example of authorization is Access control.



### Confidentiality

It is a security process that protects the leak of the data from the outsider's because it is the only way where we can make sure the security of our data.

### Authentication

The authentication process comprises confirming the individuality of a person, tracing the source of a product that is necessary to allow access to the private information or the system.



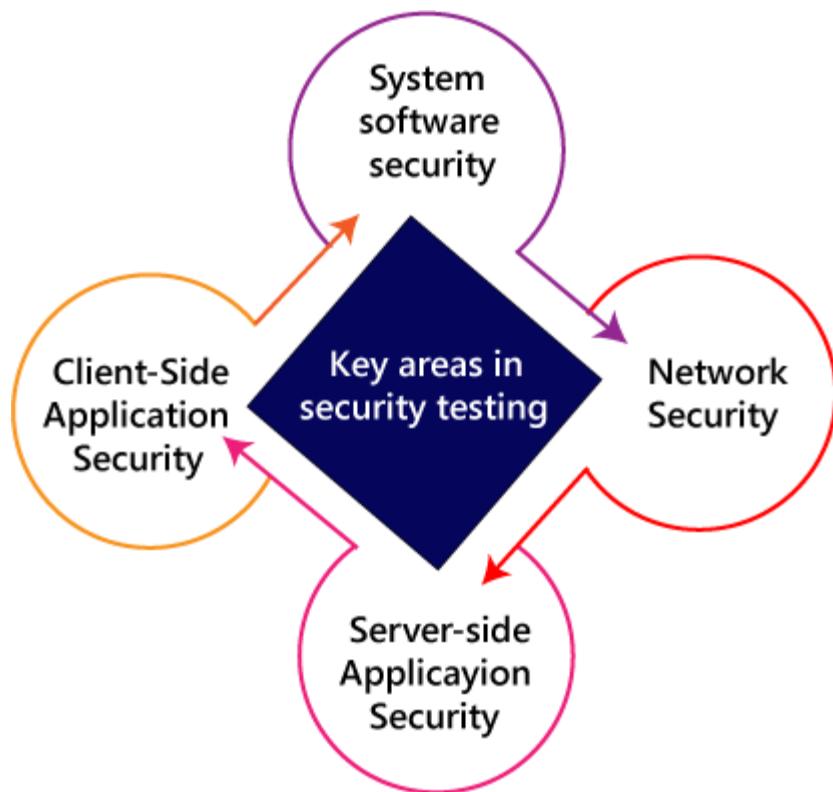
### Non-repudiation

It is used as a reference to the digital security, and it is a way of assurance that the sender of a message cannot disagree with having sent the message and that the recipient cannot repudiate having received the message.

The non-repudiation is used to ensure that a conveyed message has been sent and received by the person who claims to have sent and received the message.

### Key Areas in Security Testing

While performing the security testing on the web application, we need to concentrate on the following areas to test the application:



### System software security

In this, we will evaluate the vulnerabilities of the application based on different software such as **Operating system**, **Database system**, etc.

### Network security

In this, we will check the weakness of the network structure, such as **policies and resources**.

### Server-side application security

We will do the server-side application security to ensure that the server encryption and its tools are sufficient to protect the software from any disturbance.

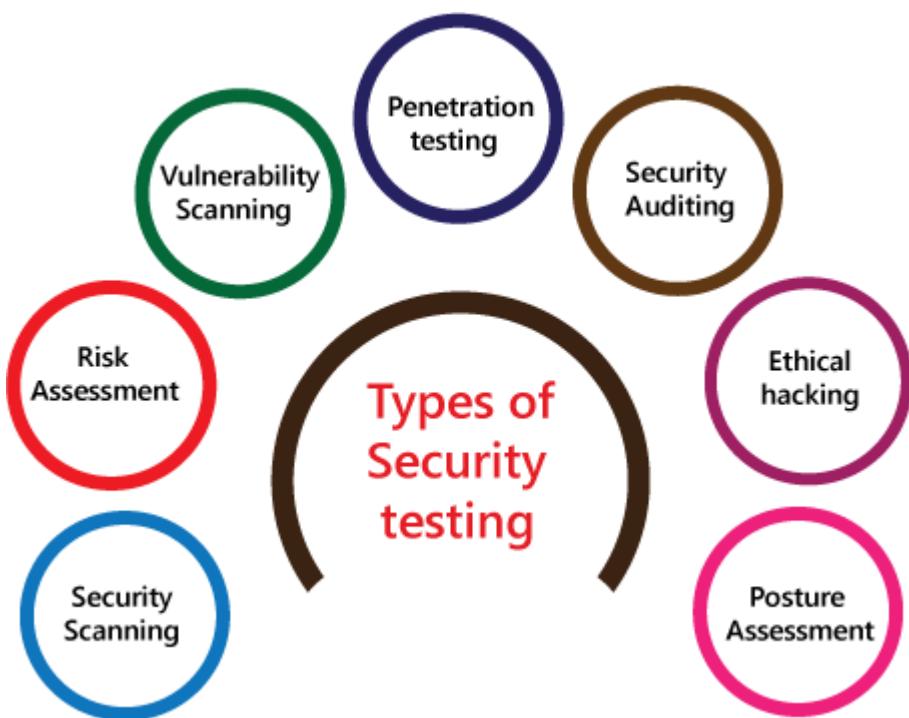
### Client-side application security

In this, we will make sure that any intruders cannot operate on any browser or any tool which is used by customers.

### Types of Security testing

As per Open Source Security Testing techniques, we have different types of security testing which as follows:

- **Security Scanning**
- **Risk Assessment**
- **Vulnerability Scanning**
- **Penetration testing**
- **Security Auditing**
- **Ethical hacking**
- **Posture Assessment**



### Security Scanning

Security scanning can be done for both **automation testing** and **manual testing**. This scanning will be used to find the vulnerability or unwanted file modification in a web-based application, websites, network, or the file system. After that, it will deliver the results which help us to decrease those threats. Security scanning is needed for those systems, which depends on the structure they use.

## Risk Assessment

To moderate the risk of an application, we will go for risk assessment. In this, we will explore the security risk, which can be detected in the association. The risk can be further divided into three parts, and those are **high, medium, and low**. The primary purpose of the risk assessment process is to assess the vulnerabilities and control the significant threat.

## Vulnerability Scanning

It is an application that is used to determine and generates a list of all the systems which contain the desktops, servers, laptops, virtual machines, printers, switches, and firewalls related to a network. The vulnerability scanning can be performed over the automated application and also identifies those software and systems which have acknowledged the security vulnerabilities.

## Penetration testing

Penetration testing is a security implementation where a **cyber-security** professional tries to identify and exploit the weakness in the computer system. The primary objective of this testing is to simulate outbreaks and also finds the loophole in the system and similarly save from the intruders who can take the benefits.

## Security Auditing

Security auditing is a structured method for evaluating the security measures of the organization. In this, we will do the inside review of the application and the **control system** for the security faults.

## Ethical hacking

**Ethical hacking** is used to discover the weakness in the system and also helps the organization to fix those security loopholes before the nasty hacker exposes them. The ethical hacking will help us to increase the security position of the association because sometimes the ethical hackers use the same tricks, tools, and techniques that nasty hackers will use, but with the approval of the official person.

The objective of ethical hacking is to enhance security and to protect the systems from malicious users' attacks.

## Posture Assessment

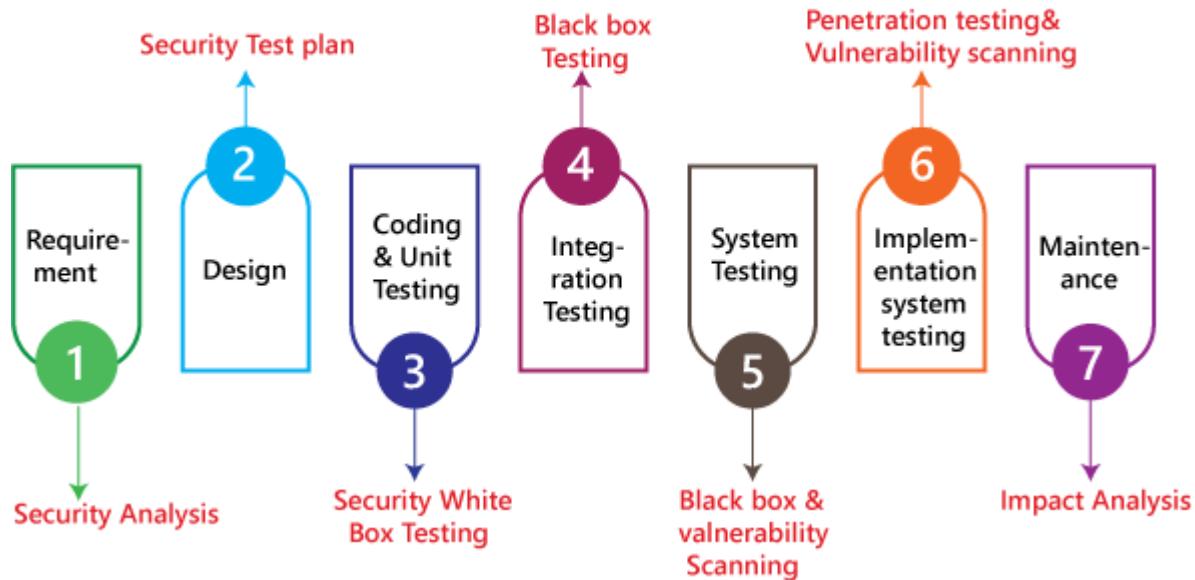
It is a combination of **ethical hacking, risk assessments, and security scanning**, which helps us to display the complete security posture of an organization.

## How we perform security testing

The security testing is needed to be done in the initial stages of the **software development life cycle** because if we perform security testing after the software execution stage and the deployment stage of the SDLC, it will cost us more.

Now let us understand how we perform security testing parallel in each stage of the software development life cycle(SDLC).

## Security Testing along with SDLC



### Step1

**SDLC:** Requirement stage

**Security Procedures:** In the requirement phase of SDLC, we will do the security analysis of the business needs and also verify that which cases are manipulative and waste.

### Step2

**SDLC:** Design stage

**Security Procedures:** In the design phase of SDLC, we will do the **security testing for risk exploration** of the design and also embraces the security tests at the development of the test plan.

### Step3

**SDLC:** Development or coding stage

**Security Procedures:** In the coding phase of SDLC, we will perform the white box testing along with static and dynamic testing.

### Step4

**SDLC:** Testing (**functional testing, integration testing, system testing**) stage

**Security Procedures:** In the testing phase of SDLC, we will do one round of **vulnerability scanning** along with black-box testing.

### Step 5

**SDLC:** Implementation stage

**Security Procedures:** In the implementation phase of SDLC, we will perform **vulnerability scanning** again and also perform one round of **penetration testing**.

### Step 6

**SDLC:** Maintenance stage

**Security Procedures:** In the Maintenance phase of SDLC, we will do the **impact analysis** of impact areas.

And the **test plan** should contain the following:

- The test data should be linked to security testing.
- For security testing, we need the test tools.
- With the help of various security tools, we can analyze several test outputs.
- Write the test scenarios or test cases that rely on security purposes.

### Example of security testing

Generally, the type of security testing includes the problematic steps based on overthinking, but sometimes the simple tests will help us to uncover the most significant security threats.

Let us see a sample example to understand how we do security testing on a web application:

- Firstly, log in to the web application.
- And then log out of the web application.
- Then click the BACK button of the browser to verify that it was asking us to log in again, or we are already logged-in the application.

### Why security testing is essential for web applications

At present, web applications are growing day by day, and most of the web application is at risk. Here we are going to discuss some common weaknesses of the web application.

- Client-side attacks
- Authentication
- Authorization
- Command execution
- Logical attacks
- Information disclosure

### Client-side attacks

The **client-side attack** means that some illegitimate implementation of the external code occurs in the web application. And the data spoofing actions have occupied the place where the user believes that the particular data acting on the web application is valid, and it does not come from an external source.

### Authentication

In this, the authentication will cover the outbreaks which aim to the web application methods of authenticating the user identity where the user account individualities will be stolen. The incomplete authentication will allow the attacker to access the functionality or sensitive data without performing the correct authentication.

**For example,** the **brute force attack**, the primary purpose of brute force attack, is to gain access to a web application. Here, the invaders will attempt n-numbers of usernames and password repeatedly until it gets in because this is the most precise way to block brute-force attacks.

After all, once they try all defined number of an incorrect password, the account will be locked automatically.

### Authorization

The authorization comes in the picture whenever some intruders are trying to retrieve the sensitive information from the web application illegally.

**For example**, a perfect example of authorization is **directory scanning**. Here the directory scanning is the kind of outbreaks that deeds the defects into the webserver to achieve the illegal access to the folders and files which are not mentioned in the public area.

And once the invaders succeed in getting access, they can download the delicate data and install the harmful software on the server.

### Command execution

The command execution is used when malicious attackers will control the web application.

### Logical attacks

The logical attacks are being used when the DoS (denial of service) outbreaks, avoid a web application from helping regular customer action and also restrict the application usage.

### Information disclosure

The information disclosures are used to show the sensitive data to the invaders, which means that it will cover bouts that planned to obtain precise information about the web application. Here the information leakage happens when a web application discloses the delicate data, like the error message or developer comments that might help the attacker for misusing the system.

**For example**, the password is passing to the server, which means that the password should be encoded while being communicated over the network.

### Note:

The web application needs more security regarding its access along with data security; that's why the web developer will make the application in such a way to protect the application from **Brute Force Attacks**, **SQL Injections**, **Session Management**, **failure to Restrict URL Access** and **Cross-site scripting (XSS)**. And also, if the web application simplifies the remote access points, then it must be protected too.

Here, **Session management**: It is used to check whether the cookies can be re-used in another computer system during the login stage.

**SQL injection**: It is a code injection approach where the destructive SQL Statements are implanted into some queries, and it is implemented by the server.

**Cross-site scripting (XSS)**: This is the technique through which the user introduces client-side script or the HTML in the user-interface of a web application and those additions are visible to other users.

### Myths and Facts of Security testing

| Myths                                                                                                                           | Facts                                                                                                                                                         |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| There is no profit on an asset in security testing.                                                                             | Security Testing can highlight the parts of enhancement, which help us to increase productivity and decrease interruption and also allow the maximum output.  |
| Nowadays, the Internet is not safe, and we will be buying the hardware or software to save the business and protect the system. | Here, the fact is rather purchasing any software or hardware, the company first understands security and then follows the security process.                   |
| If we have a small business and we do not require a security procedure.                                                         | In that case, the fact is every organization needs a security procedure.                                                                                      |
| The only technique to secure is to disconnect it.                                                                               | The one and the most excellent way to protect an association is to identify the Perfect Security. Here Perfect security can be accomplished by performing the |

implementation, compared with business, permitted, and manufacturing validations.

## Testing in the Agile Environment

**Agile Testing** is a testing practice that follows the rules and principles of agile software development. Unlike the Waterfall method, Agile Testing can begin at the start of the project with continuous integration between development and testing. Agile Testing methodology is not sequential (in the sense it's executed only after coding phase) but continuous.

### Principles of Agile Testing

Here are the essential Principles of Agile Testing:

- In this Agile testing model, working software is the primary measure of progress.
- The best result can be achieved by the self-organizing teams.
- Delivering valuable software early and continuously is our highest priority.
- Software developers must act to gather daily throughout the project.
- Enhancing agility through continuous technical improvement and good design.
- Agile Testing ensures that the final product meets the business's expectations by providing continual feedback.
- In the Agile Test process, we need to execute the testing process during the implementation, which reduces the development time.
- Testing process in Agile should work on the consistent development pace
- Provide regular reflections on how to become more effective.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- Each time the team meets, it reviews and adjusts its behavior in order to become more effective.
- Face-to-face conversation with the development team is the most effective and efficient method of conveying information within the team.

Agile Testing includes various principles that help us increase the Software's productivity.

### Agile Testing Life Cycle

The agile Testing life cycle is completed in five different phases, as we can see in the following image:



Here are the Agile process testing steps:

**Phase1: Impact Assessment:** In this initial phase, we gather inputs from stakeholders and users. This phase is also called the feedback phase, as it assists the test engineers in setting the objectives for the next life cycle.

**Phase 2: Agile Testing Planning:** It is the second phase of the Agile testing life cycle, where all stakeholders come together to plan the schedule of the testing process and deliverables.

**Phase 3: Release Readiness:** At this stage, we review the features that have been developed/ Implemented are ready to go live or not. In this stage, it is also decided which one needs to go back to the previous development phase.

**Phase 4: Daily Scrums:** This stage includes every standup morning meeting to catch up on the status of testing and set the goal for the entire day.

**Phase 5: Test Agility Review:** The last phase of the Agile life cycle is the Agility Review Meeting. It involves weekly meetings with stakeholders to regularly evaluate and assess progress against goals.

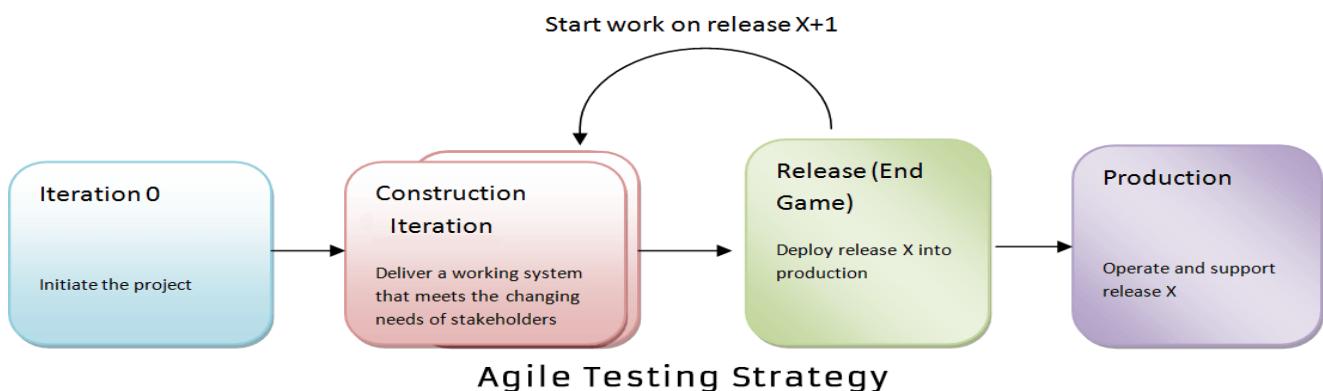
## Agile Test Plan

**Agile test plan** includes types of testing done in that iteration like test data requirements, infrastructure, test environments, and test results. Unlike the waterfall model, in an agile model, a test plan is written and updated for every release. Typical test plans in agile includes

- Testing Scope
- New functionalities which are being tested
- Level or Types of testing based on the features complexity
- Load and Performance Testing
- Infrastructure Consideration
- Mitigation or Risks Plan
- Resourcing
- Deliverables and Milestones

## Agile Testing Strategies

Agile testing life cycle spans through four stages



### Iteration 0

During the first stage or iteration 0, you perform initial setup tasks. It includes identifying people for testing, installing testing tools, scheduling resources (usability testing lab), etc. The following steps are set to achieve in Iteration 0

- Establishing a business case for the project
- Establish the boundary conditions and the project scope
- Outline the key requirements and use cases that will drive the design trade-offs
- Outline one or more candidate architectures
- Identifying the risk
- Cost estimation and prepare a preliminary project

### Construction Iterations

The second phase of agile testing methodology is Construction Iterations, the majority of the testing occurs during this phase. This phase is observed as a set of iterations to build an increment of the solution. In order to do that, within each iteration, **the team implements** a hybrid of practices from XP, Scrum, Agile modeling, and agile data and so on.

In construction iteration, the agile team follows the prioritized requirement practice: With each iteration, they take the most essential requirements remaining from the work item stack and implement them.

Construction iteration is classified into two, confirmatory testing and investigative testing. **Confirmatory testing concentrates** on verifying that the system fulfills the intent of the stakeholders as described to the team to date, and is performed by the team. While the investigative testing detects the problem that confirmatory team has skipped or ignored. In Investigative testing, tester determines the potential problems in the form of defect stories. Investigative testing deals with common issues like integration testing, load/stress testing, and security testing.

Again for, confirmatory testing there are two aspects **developer testing** and **agile acceptance testing**. **Both of them** are automated to enable continuous regression testing throughout the lifecycle. Confirmatory testing is the agile equivalent of testing to the specification.

Agile acceptance testing is a combination of traditional functional testing and traditional acceptance testing as the development team, and stakeholders are doing it together. While developer testing is a mix of traditional unit testing and traditional service integration testing. Developer testing verifies both the application code and the database schema.

### **Release End Game Or Transition Phase**

The goal of "Release, End Game" is to deploy your system successfully into production. The activities include in this phase are training of end users, support people and operational people. Also, it includes marketing of the product release, back-up & restoration, finalization of system and user documentation.

The final agile methodology testing stage includes full system testing and acceptance testing. In accordance to finish your final testing stage without any obstacles, you should have to test the product more rigorously while it is in construction iterations. During the end game, testers will be working on its defect stories.

### **Testing Web and Mobile Applications**

As we know, explosive growth in the use of mobile devices and the development of mobile devices makes testing an essential requirement for the successful and rapid delivery of high-quality mobile applications.

### Why is mobile testing Important?



As we know in the current time, the use of mobile phones is increasing day by day. People install new applications based on ratings and reviews daily. The ratings and reviews of the applications depend on the working of the application. So, this thing makes the testing of the mobile application important. Different mobile phones run on different operating systems and contain different screen sizes; so, this thing makes the mobile phone's testing is a mandatory process in the software development process. The updating in the app entertain the users. New bug fixes in the app makes sure that nobody uninstalls the app. Testing is essential for apps to stay in the market.

As we know in today's time, mobile phones are popular in the market. We do not want to switch ON our PCs and Laptops. Instead, we want those devices which can be easily handled and can perform the task quickly.

That is why mobile phones should be adequately tested before releasing them to the market.

Backward Skip 10s Play Video Forward Skip 10s

### Approaches to Test the mobile application

We will follow two different approaches for the testing of mobile applications. Here we will follow two approaches to test the mobile application, and those are manual testing and automated testing.

#### Manual Testing



**Manual testing** is a human process. The primary focus of manual testing is on the experience of the user. The Analysis and evaluation of the application's functionality can be done through the medium of the user in an explorative process. Manual Testing ensures that the application work on the standard of user-friendliness. Manual testing is generally the time-consuming process because the process is to find out the bugs will take time. Therefore, according to the thumb rule, 20% of applications at the time of releases should be tested with the alpha and beta testing. In the rest part of the application, automated testing should be performed.

## Automation Testing



Automated testing is the second approach to test the mobile application. In this process, an array of test cases is set up. Automated Testing covers 80% of the testing process. The percentage is not set up, but this is the general guideline followed by the software industry. We will perform **automation testing** in the below scenarios-

- When manual test cases are slow, then we will use Automate test cases.
- The test cases which can be easily automated then we will use Automation Testing.
- Automation testing is used for test cases, which is written for the frequently used functionality.
- Automation testing is used for the automated test cases, which we cannot perform manually.
- Automation testing is used to automate the test cases, which gives us predictable results.

## Advantages of Mobile Testing Automation

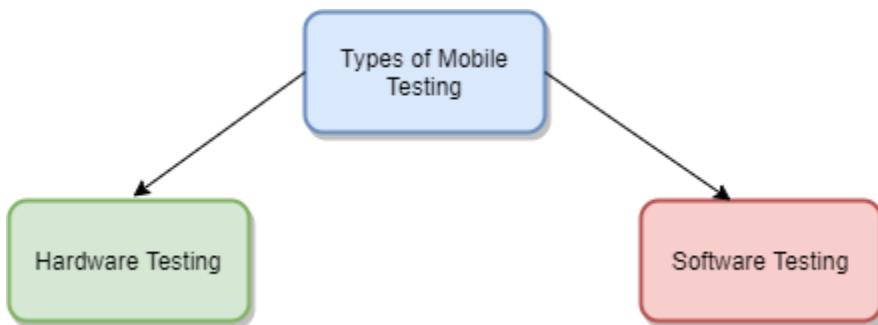
The Automation testing of the mobile application is very useful. The advantages of the Automation testing are just like as below:

- Automation Testing increases the efficiency of testing.

- Automation testing enhances the regression test case execution.
- The use of the automation testing for the testing of the mobile application saves the bunch of time, and also can execute more test cases.
- In the automation testing, we can perform the same test scripts again and again
- Multiple devices can run the test script parallelly.

### Types of Mobile Testing

Mobile testing has two types that we can perform on mobile devices.

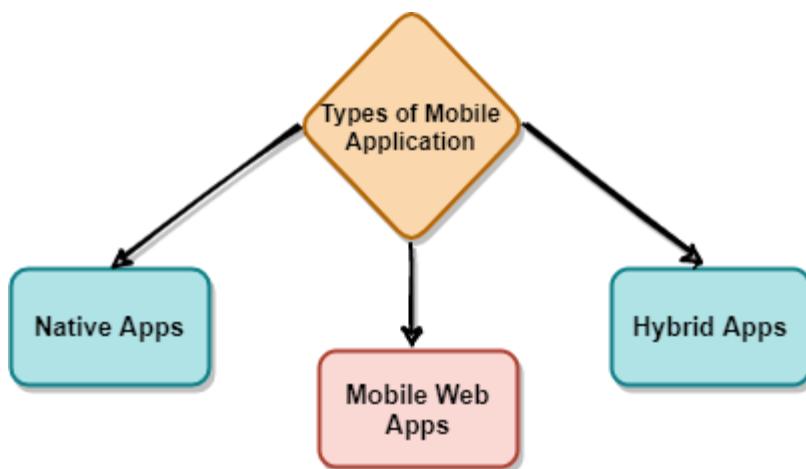


**1. Hardware Testing:** Hardware testing is done on the internal processors, resolution, internal hardware, size of the screen, radio, space, camera, Bluetooth or WIFI, etc. This testing is known as simple "mobile testing."

**2. Software Testing or the Application Testing:** The functionality of those applications should be tested, which works on mobile devices. We call this testing as the "Mobile Application Testing".

### Types of the Mobile Application

There are three types of applications. These are as shown below:



**Fig: Types of Mobile Application**

- a. **Native Apps:** Native application is used on different platforms like mobile and tablets.
- b. **Mobile Web Apps:** Mobile apps are the server-side apps. We can access the mobile web apps on mobile by using the different browsers like Chrome, Firefox, after connecting the mobile to the mobile network or wireless network like WIFI.
- c. **Hybrid Apps:** Hybrid applications are a combination of both types of applications like native and web applications. Hybrid apps run offline on the devices. Hybrid apps are written by using the web technologies like HTML5 and CSS.

The differences between these applications are:

1. Native applications are dependent on a single platform; on the other hand, mobile applications are the cross-platform applications.
2. Native apps are performed on the SDKs platform, and the Mobile web apps used web technologies like HTML, CSS, asp.net, Java, and PHP.
3. Native applications need the installation while the mobile web apps do not require any installation.
4. The updation in the mobile native apps can be done from the play store or app store. On the other hand, mobile web apps updations are centralized.
5. There is no requirement for the internet connection for native apps, but mobile web apps need an internet connection.

6. Native apps work faster than mobile web apps.
7. Native application can be installed from app stores like Google play store or app store where mobile web are websites and are only accessible through the Internet.

### Significance of Mobile Testing

Testing the applications on mobile devices is a more difficult task than testing web applications. This is just because of the below reasons:

- **Mobile device's different range:** This is one reason why the application testing is challenging to do on mobile devices. There are different ranges of devices available in the market that have different screen sizes and hardware configurations like a hard keypad, virtual keypad (touch screen) and trackball, etc.
- **Huge varieties of mobile devices:** There are many mobile devices available in the market like HTC, Samsung, Apple, and Nokia. That is why the testing of mobile devices is challenging.
- **Different mobile operating systems:** Different mobile devices use different operating systems like Android, Symbian, Windows, Blackberry, and IOS. This is also the main reason why mobile testing is difficult.
- **Different versions of operation system:** In the market, there are lots of different versions of the operating system available like iOS 5.x, iOS 6.x, BB5.x, BB6.x etc.
- **Different mobile network operators:** Mobile contains different network operators like GSM and CDMA.
- **Frequent updates** - The frequent update in the mobile phones (like Android- 4.2, 4.3, 4.4, iOS-5.x, 6.x) make sure that the new testing cycle should not affect the functionality of the application.

Mobile application testing is essential because the millions of users can use one specific product. If there is a bug in the product, that product will not be accepted by the client. The bug in the product can be the loss of memory, legal issue, and the irreplaceable damage in the image.

### Differences between the Mobile and the Desktop application testing

- The desktop uses the central processing unit to test the applications. And the mobile device tests the application on handsets like Samsung, Nokia, Apple, and HTC.
- The screen size of the mobile application is smaller than the desktop application.
- The memory of the mobile device is less than the desktop.
- Network connections used for mobile applications are like 2G, 3G, 4G, or WIFI, whereas the desktop application use broadband or dial-up connections.
- There can be the chances that the tool used for the testing of the desktop application will not work to test the mobile application.

### Types of the Mobile Application Testing

The following are the types of testing that can be performed on mobile:

- **Usability Testing:** Usability testing ensures the user that the mobile app is easy to use and gives a satisfactory experience to the user.
- **Compatibility Testing:** Compatibility testing is done on different mobile devices, on different browsers, screen sizes, and the versions of the Operating system according to the requirement.
- **Interface Testing:** Interface testing is done on the menu options, bookmarks, buttons, history, settings, and the navigation flow of the application.
- **Service Testing:** Service Testing is done to test the services of the application online or offline.
- **Low-Level resource Testing:** This type of testing is done on the memory usage, auto-detection of the temporary file, growing issue of the local database. The testing of all the resources is known as Low-Level Resource Testing.
- **Performance Testing:** Performance testing is done on the application after changing the connection from 2G,3G to WiFi. We will test the performance of the application by sharing the documents, battery consumption, etc.
- **Operational Testing:** Operational Testing will perform to test the backup and recovery plan when the battery goes down or the loss of the data when we upgrade the application from the store.

- **Installation Testing:** Installation Testing will be performed to validate the application by installing and uninstalling the app on the device.
- **Security Testing:** Security testing is done to test the application to validate the information to protect it.

### Mobile Application Testing Strategies

While deciding the testing strategy, we should have to ensure that the quality and the performance guidelines met. Points are as shown below:

- a. **Selection of the device:** During the selection of the device in the first analyses the market and choose those devices which are widely used these days. The selection of the device depends on the client.
- b. **Emulators:** The use of the emulator is useful in the initial stage of the development; the emulator allows us to test the application quickly. An emulator is a system used to run the software from one environment to another environment without any change in the software system. The emulator works on the real system.

### Type of the Mobile Emulator

- **Device Emulator:** The manufacturer of the device provides this emulator.
- **Browser Emulator:** This emulator simulates the environment of the mobile browser.
- **Operating Systems Emulator:** The operating system Apple provides us the emulator for the iPhones, Google provides the emulator for the Android Phone, and the Microsoft provides the emulator for Windows Phone.

The usage of mobile application is increasing in all business areas like Schools, Publishers, Retailers, HealthCare providers, etc. This evolves the use of the mobile application that increases the businesses focus to develop a robust mobile application and the testing of the mobile application before it is launched in the market. Only the development of the mobile application is not enough; it also needs testing before introducing the app into the market. The testing is essential so that the application can run smoothly without any difficulties.

## Mobile Testing Process

*The steps which involve in the process of Mobile Testing are as shown below:*

**Step 1) Planning and Preparation:** When the development stage is complete, we should have to identify the app's objective and the limitation of the app.

***Planning and Preparation involve the below steps:***

1. First, find out the functional Requirement.

***Now we will find out the following requirements as shown below:***

- a. In the first, we should know whether the application is working together with the other apps.
- b. Now we will check about the apps, whether it is Native, Hybrid, or Mobile-web.
- c. Does the app require the front-end or the back-end testing?
- d. Do all the features of the apps working correctly?
- e. How the app controls the load.
- f. The changed status of the phone has any effect on the application?
- g. We have to be sure that the testing team is ready to execute the process.

### **Step 2) Classify the types of testing.**

In this step, we will identify the types of testing which we want to apply to mobile applications. We will do the usability, compatibility, performance, and security testing on the target devices. And we will also define which functionality should be tested.

### **Step 3) Preparation of the test case and the script design.**

Here we will create the test case for every function of the application.

Prepare the document of the test case for every feature and functionality.

In addition to functional test cases, some exceptional cases should also be covered:

It is essential to know which testing we will use, either it is manual or automation.

Prepare separate suite for manual test cases and automated test scripts as per the requirement. Identify the reusable automation scripts and modify them as per the requirement of the project.

### **Step 4) Environment Set up for Testing**

For environment setup, we will follow the below steps:

**Step 1. Identify the types of testing:** We know that the ILL application is applicable for browsers, so we have to test the application on all the supported browsers by using different mobile devices. Different testing like **usability**, **functional**, and **compatibility** testing will do on the different browsers by using **manual** and **automation** test cases.

**Step 2. Manual and Automated testing:** The method we used here to test the mobile application is the Agile testing. The developers will release the new product for the testing team, and the testing team runs their test cases in the environment of Quality Assurance. The team of the automation creates scripts for the set of the basic functionalities and runs the scripts, which help us to determine that the new build is stable enough to test?the team of **Manual Testing** tests the new functionality.

**JIRA** is used to write about the acceptance criteria, maintenance of test cases, and logging /re-verification of defects. When the iteration is complete, a meeting will be held for the **iteration planning** between the developers, team, product owner, business analyst, and QA team. They will discuss **what went well** and **what needs to improve**.

**Step 3. Beta Testing:** After the completion of the regression testing, now the team of the Quality Assurance will do the User **Acceptance Testing**. The client does user Acceptance Testing. They re-verify all the bugs to ensure that bug was fixed, and the application is working as expected on every approved browser.

**Step 4. Performance test:** The performance of the Web Application can be tested by using the JMeter, and also by increasing the loads on the web applications can also test the performance of the applications.

**Step 5. Browser testing:** **Browser testing** can be done using simulation tools and using real mobile devices to test web applications over multiple browsers.

**Step 6. Launch plan:** In the last phase, testing moves into staging; at the end of the 4<sup>th</sup>-week testing team will test the application, which is the final round of testing where the testing is done to be sure that the product is ready for the production. After the completion of this testing, now the software goes Live!

### Web testing

Web testing is a software testing technique to test web applications or websites for finding errors and bugs. A web application must be tested properly before it goes to the end-users. Also, testing a web application does not only means finding common bugs or errors but also testing the quality-related risks associated with the application. **Software Testing** should be done with proper tools and resources and

should be done effectively. We should know the architecture and key areas of a web application to effectively plan and execute the testing.

Testing a web application is very common while testing any other application like testing functionality, configuration, or compatibility, etc. Testing a web application includes the analysis of the web fault compared to the general software faults. Web applications are required to be tested on different browsers and platforms so that we can identify the areas that need special focus while testing a web application.

### **Types of Web Testing:**

Basically, there are 4 types of web-based testing that are available and all four of them are discussed below:

- **Static Website Testing:** A static website is a type of website in which the content shown or displayed is exactly the same as it is stored in the server. This type of website has great UI but does not have any dynamic feature that a user or visitor can use. In static testing, we generally focus on testing things like UI as it is the most important part of a static website. We check things font size, color, spacing, etc. testing also includes checking the contact us form, verifying URLs or links that are used in the website, etc.
- **Dynamic Website Testing:** A dynamic website is a type of website that consists of both frontend i.e, UI, and the backend of the website like a database, etc. This type of website gets updated or change regularly as per the user's requirements. In this website, there are a lot of functionalities involved like what a button will do if it is pressed, are error messages are shown properly at their defined time, etc. We check if the backend is working properly or not, like does entering the data or information in the GUI or frontend gets updated in the databases or not.
- **E-Commerce Website Testing:** An e-commerce website is very difficult in maintaining as it consists of different pages and functionalities, etc. In this testing, the tester or developer has to check various things like checking if the shopping cart is working as per the requirements or not, are user registration or login functionality is also working properly or not, etc. The most important thing in this

testing is that does a user can successfully do payment or not and if the website is secured. And there are a lot of things that a tester needs to test apart from the given things.

- **Mobile-Based Web Testing:** In this testing, the developer or tester basically checks the website compatibility on different devices and generally on mobile devices because many of the users open the website on their mobile devices. So, keeping that thing in mind, we must check that the site is responsive on all devices or platforms.

### **Points to be Considered While Testing a Website:**

As the website consists of a frontend, backend, and servers, so things like HTML pages, internet protocols, firewalls, and other applications running on the servers should be considered while testing a website. There are various examples of considerations that need to be checked while testing a web application. Some of them are:

- Do all pages are having valid internal and external links or URLs?
- Whether the website is working as per the system compatibility?
- As per the user interface- Does the size of displays are the optimal and the best fit for the website?
- What type of security does the website need (if unsecured)?
- What are the requirements for getting the website analytics, and also controlling graphics, URLs, etc?
- The contact us or customer assistance feature should be added or not on the page, and etc?

### **Steps in Software Testing**

There is a total of 11 steps in software testing. You can read all of them from the article [General Steps of Software Testing Process](#). In **web-based testing**, various areas have to be tested for finding the potential errors and bugs, and **steps for testing a web app are given below:**

- **App Functionality:** In web-based testing, we have to check the specified functionality, features, and operational behavior of a web application to ensure they

correspond to its specifications. For example, Testing all the mandatory fields, Testing the asterisk sign should display for all the mandatory fields, Testing the system should not display the error message for optional fields, and also links like external linking, internal linking, anchor links, and mailing links should be checked properly and checked if there's any damaged link, so that should be removed. We can do testing with the help of [Functional Testing](#) in which we test the app's functional requirements and specifications.

- **Usability:** While testing usability, the developers face issues with scalability and interactivity. As different numbers of users will be using the website, it is the responsibility of developers to make a group for testing the application across different browsers by using different hardware. For example, Whenever the user browses an online shopping website, several questions may come to his/her mind like, checking the credibility of the website, testing whether the shipping charges are applicable, etc.
- **Browser Compatibility:** For checking the compatibility of the website to work the same in different browsers we test the web application to check whether the content that is on the website is being displayed correctly across all the browsers or not.
- **Security:** Security plays an important role in every website that is available on the internet. As a part of security, the testers check things like testing the unauthorized access to secure pages should not be permitted, files that are confined to the users should not be downloadable without the proper access.
- **Load Issues:** We perform this testing to check the behavior of the system under a specific load so that we can measure some important transactions and the load on the database, the application server, etc. are also monitored.

- **Storage and Database:** Testing the storage or the database of any web application is also an important component and we must sure that the database is properly tested. We test things like finding errors while executing any DB queries, checking the response time of a\the query, testing whether the data retrieved from the database is correctly shown on the website or not.

**UNIT V TEST AUTOMATION AND TOOLS****6**

Automated Software Testing, Automate Testing of Web Applications, Selenium: Introducing Web Driver and Web Elements, Locating Web Elements, Actions on Web Elements, Different Web Drivers, Understanding Web Driver Events, Testing: Understanding Testing.xml, Adding Classes, Packages, Methods to Test, Test Reports

**Automated Software Testing**

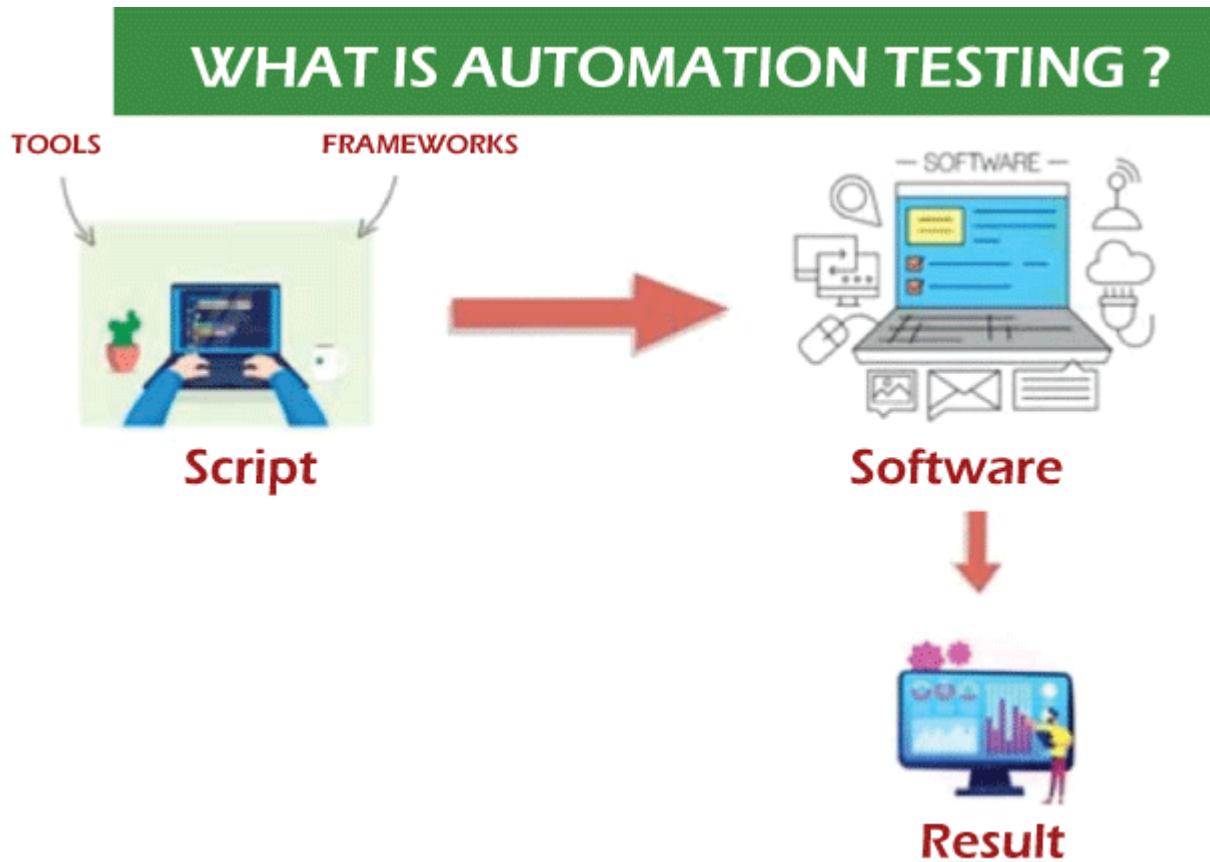
Software is the medium or platform for performing a task in the system. Software is developed by following the SDLC process, i.e. software development life cycle in which software has to pass through many phases, and testing is one of its major phase that shows the result of all the efforts made in developing process of software upto now.



# What is Automation Testing

Software testing is not a one-time process, as the developer test the software until it is declared error-free. It checks whether the software is showing the expected output per input. Software testing is done by the testers who test cases repeatedly to gain output according to the requirement in two ways: **a) manually and b) automatically.**

Every project or software goes through a testing procedure, and the type of testing method depends on various factors like the budget, expertise, suitability, requirement and timeline of the project. When the tester writes the testing script manually and tests the software until it functions properly known as manual testing & when this manual testing process becomes automatic, it can be defined as **automatic testing**.



Manually when testers write the test cases and execute them repeatedly, known as manual testing, which is time-consuming and the test results are not sure. So to recover these drawbacks, automation testing came into existence. With automation testing, developers or testers keep everything in their hands, boosting the product's productivity.

### Advertisement

Manual testing is best suitable for usability testing, Ad-hoc testing and usability testing. In contrast, automation testing is best suited for areas like load testing, regression testing, repeated execution and performance testing. Automation helps companies take new features to market instantly in the testing world and ensure a bug-free user experience.

***"Automation testing refers to the automatic testing of the software in which developer or tester write the test script once with the help of testing tools and framework and run it on the software. The test script automatically test the software without human intervention and shows the result (either error, bugs are present or software is free from them)."***

Automation testing needs manual effort when creating initial scripts, and further process is performed automatically to compare the actual testing result with expected results. This can be performed at these levels a) unit-level automation, b) API testing c) user interface.

### Manual testing vs. Automation testing

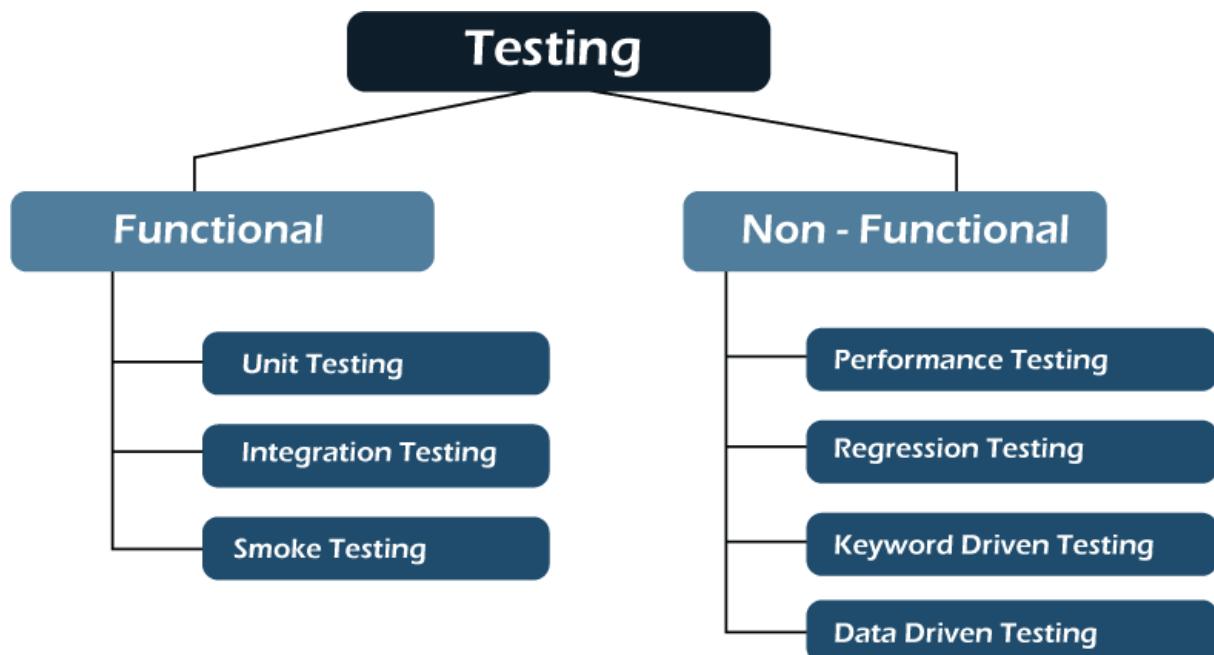
Software testing is performed to discover bugs in software during its development. The key difference between automation and manual testing are as follows:

| Manual testing                                                                                                   | Automation testing                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Testing in which a human tester executes test cases                                                              | In automation testing, automation tools are used to execute the test cases                                                                   |
| In this testing, human resources are involved, that's why it is time-consuming                                   | It is much faster than the manual testing                                                                                                    |
| It is repetitive and error-prone                                                                                 | Here automated tools are used that make it interesting and accurate                                                                          |
| BVT (build verification testing) is time-consuming and tough in manual testing                                   | It's easy to build verification testing                                                                                                      |
| Instead of frameworks, this testing use checklist, guidelines, and stringent process for drafting test cases.    | Frameworks like keyword, hybrid, and data drive to accelerate the automation process.                                                        |
| The process turnaround time is higher than the automation testing process (one testing cycle takes lots of time) | It completes a single round of testing within record time; therefore, a process turnaround time is much lower than a manual testing process. |
| The main goal of manual testing is user-friendliness or improved customer experience.                            | Automation testing can only guarantee a positive customer experience and user-friendliness.                                                  |

|                                                         |                                                                                |
|---------------------------------------------------------|--------------------------------------------------------------------------------|
| It is best for usability, exploratory and adhoc testing | It is widely used for performing testing, load testing and regression testing. |
| Low return on investment                                | The high return on investment                                                  |

### Automation testing types

Testing is grouped under two types: *functional and non-functional*



### Functional testing

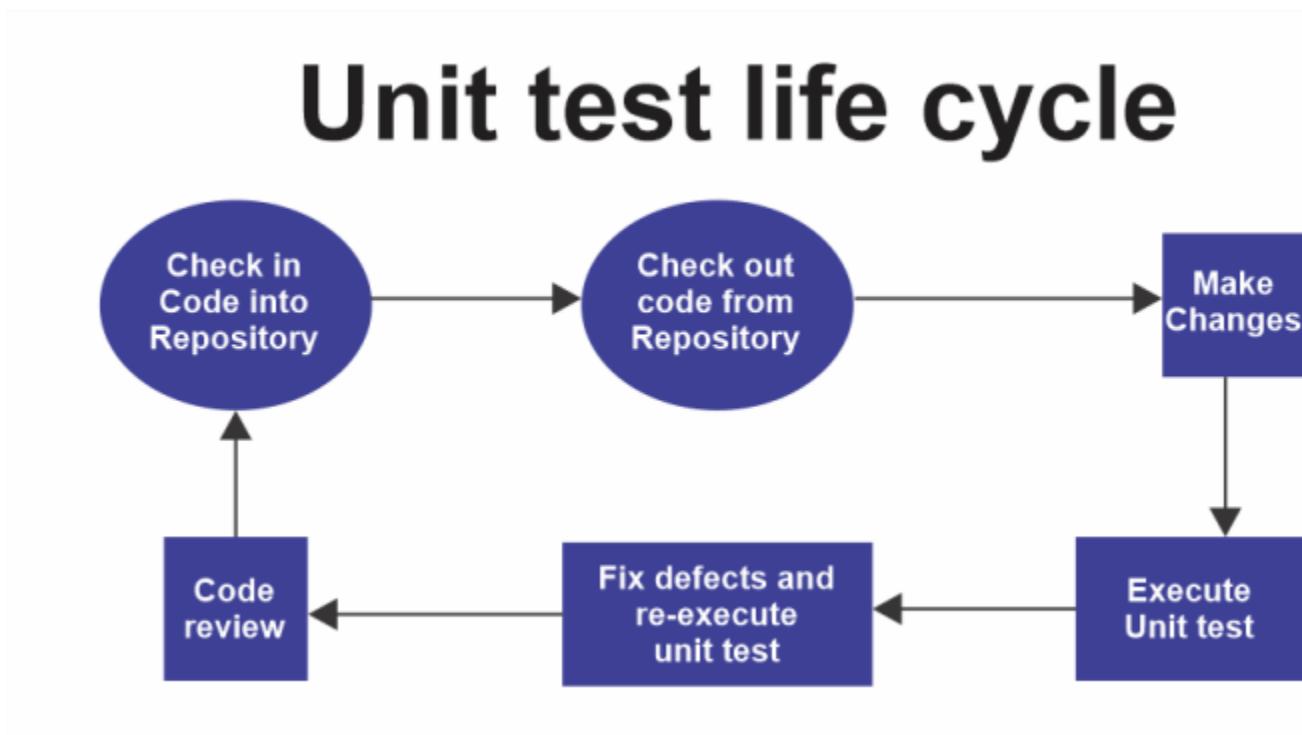
The first test performed by tester on newly revised software is called functional testing, which verifies all the software functions' features per user requirement. This testing works on the real-world business application and obtaining the expected output from a given input. All application functions are tested and involve smoke, unit, and integration testing.

#### a) Unit testing

The unit is the smallest component of the software that functions individually. Unit testing simplifies the testing of the whole software, where each software element is fully

tested before the final version is out. Unit testing depicts how the code performs at each part and has a faster execution time.

It's the favourite of developers because it consumes less time and assure the working of each part of the software. Before automation testing, the developers write the code for testing, but now there is no need. The unit testing technique is divided into three broad categories: White box testing, Black box testing and Grey box testing.

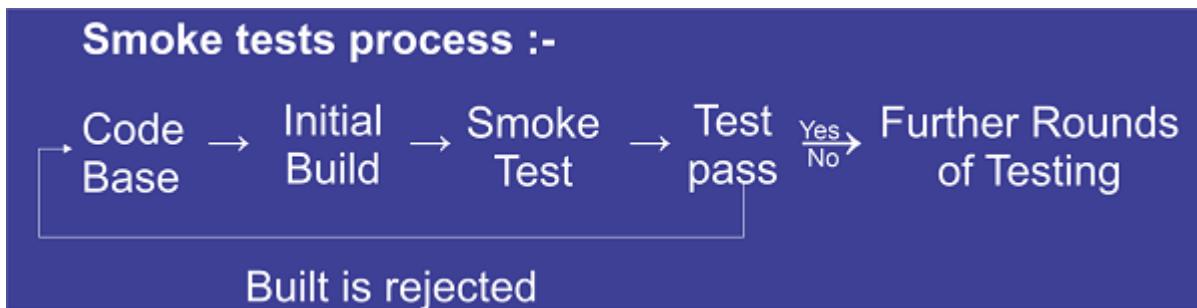


### b) Integration testing

Integration testing is more complicated to set up compared with other tests. All the modules of the application communicate with each other to perform tasks. Therefore, testers group them for testing and exposing the flaws in maintaining the interaction between these modules. Another name for this testing is I&T or string testing, considered end-to-end.

### c) Smoke testing

This testing checks and defines the product's stability (whether stable or not). If the product result is unstable, it is called an 'unstable build' and sent back to developers, where they run more test cases to find out the root cause of the problem. The smoke test works like this:-



### Non-functional testing

Non-functional testing focuses on how well application functions are doing, not on what the product does. It is the opposite of functional testing, where application elements like reliability, usability, performance, etc., are tested. Some types of non-functional testing are reliability testing, load testing, compatibility testing, performance testing, security testing etc.

#### a) Performance testing

This non-functional testing tests the software's stability, responsiveness and speed under the workload. It finds out the potential issues faced by critical software and medical programs used by the user, like slow operation of software under stressful circumstances. It finds hurdles in the performance of software and removes them to increase the ability of software to deliver the best results to the end user.

#### b) Regression testing

When some changes are made to the code of software or application, it needs to be tested to determine whether the software is working as before the change; for this purpose, testers use automation regression testing to automate scripts, applications of workflows, plans and other activities. It tests the system or software workflow after its updation and functional error.

#### c) Keyword driven testing

Keyword-driven testing tests the application using the data files consisting of the keywords related to the application, representing a set of actions needed to carry out the step. Here these specific keywords are identified and connected with the specific action. Therefore during testing, when these keywords are used, their related actions will automatically be done. This keyword testing is a popular choice for many businesses as it's flexible, concise, easy to maintain and reusable. Keyword-driven testing is compatible with all kinds of automation tools in the market. Instead of programming experts, functional testers can plan the testing before the application is fully developed.

#### d) Data-driven testing

In data-driven testing, automation is inbuilt and very effective due to the few facilities provided, like the reusability of code, change in the script doesn't affect the test cases, and this testing can be carried out in the phase of the software development cycle. It provides consistency in results and reduces the investment of time and resources. Test cases use the data separately stored in the table or spreadsheet format, and testers have multiple data sets for testing.

#### Automation testing process

There are certain steps followed in the automation testing process:

##### **Step1: To convince the management**

→ In the business, testing is not only in the hand of the tester because automation testing tools are expensive, and large licence fees are associated with them. A tester alone can't do it; they need to convince the management about the benefits test automation offers.

As we know, automation is a little expensive, so after doing a cost-benefit analysis, we need to convince management by preparing a detailed report about the benefit of test automation.

They have to convenience that automation test results can't be seen immediately. It takes 2-3 months to show results. You have to convenience yourself that it's a matter of patience to achieve an error-free application.

##### **Step 2: Recruitment of Automation tool expert**

→ Automation testing is done by automation tool experts who know how to use them for better results. There are two types: Automation architectures and automation engineers.

- **Automation architecture** builds the automation framework, creates rules for scripting & designs the naming convention. They help the management analyze applications and tools used in the application so that they can select the right tool for automation. They are experienced in using different kinds of tools as they understand them well with their pros and cons, so they can select the test case that needs to be automated.
- **Automation engineers** are experts in converting manual test cases into automated test scripts and work under the automation architect. Engineers must be good in programming (object-oriented language), which helps them

understand, create & execute the scripts. Companies can hire them from outside or prepare them inside by training their existing manual testers.

### **Step 3: Selection of the right automation tool**

→ This is the most important and challenging step in the testing process because the wrong tool gives you results that can harm the business. Before selecting an automation tool, one must know its pros and cons.

After that, business needs and requirements are considered and analyzed before making the final decision. Points considered in selecting the tool: The tool must lie in your budget, support the technologies used in the application, skilled resources are necessary, and tools have a good reporting mechanism.

### **Step 4: Choosing the application**

→ For automation testing, it is important to select the right tests for automation with the right application, and it depends on certain factors like:

- a. The selected application must be bug-free after manual testing
- b. Application UI must not change frequently and must be stable
- c. Selected application for automation must be in the early stage of its development with stable modules and manually tested results

### **Step 5: Automation teams are trained**

→ With the technological change, manual testers need to be upgraded to become automation engineers. They must be trained in automation concepts and terminologies. The automation testing team must be trained to use the tool for testing and giving the expected results.

### **Step 6: Automation testing framework is developed**

→ After selecting tools, applications and training for the automation team, it's time to develop the framework for the automation testing. The framework combines planning strategies and rules to write the test script. This could lead to the least amount of maintenance in the application, and if there is a need for any change that it's easy to deal with. It includes modular, data-driven, hybrid, keyword driven and linear frameworks.

### **Step 7: Preparing an execution plan**

→ The automation testing is executed with an execution plan involving selecting an environment on which the script will execute, including browser, operating system and other hardware configurations. When writing the script, which configuration will be executed? The automation team executes the plan and states who will execute the scripts. The execution plan varies from company to company as some ask its developers to execute the plan before release, and some hire a direct resource.

### **Step 8: Writing of scripts**

→ When the ground-level work is complete (including framework designing, execution of the plan, and resources trained on the new tool), the time comes to write the scripts. Points should be remembered while writing the script:

- a. Code loss should be prevented by preserving it in source control
- b. In source code history and version control should be preserved
- c. Every script should be maintained in a formulated manner (proper naming conventions)
- d. The scriptwriter must know programming languages because automation testing is a part of software development

### **Step 9: Reporting**

→ Reporting includes the result of the testing created at the end of the execution. It consists of all the detailing of the testing and what is yet remaining. The result is written in tables and charts according to the management requirement and mailed to management.

### **Step 10: Script maintenance**

→ Script maintenance step is required when a change is requested in the application. To cope with the changes, scripts are immediately updated to ensure flawless execution. Regular script maintenance is important for the smooth running of the application and bug-free operation.

### **Automation testing tools**

Automation testing is conducted by using automation tools. Selecting the appropriate tool for testing is quite difficult and important as the success of an automation test depends on it. Here are some automation testing tools used by testers nowadays:

**Selenium:**

The screenshot shows the official Selenium website at [selenium.dev](https://selenium.dev). The header features the Selenium logo and navigation links for About, Downloads, Documentation, Projects, Support, Blog, and English. A search bar is also present. The main banner has a green background with the text "Selenium automates browsers. That's it!" and "What you do with that power is entirely up to you." Below the banner, a note states: "Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should) also be automated as well." Three main sections are highlighted: "Getting Started" with icons for Selenium WebDriver (red square with white 'Se' and checkmark), Selenium IDE (blue square with white 'Se' and video camera icon), and Selenium Grid (blue square with white 'Se' and network icon). Each section includes a brief description: "If you want to create robust," for WebDriver, "If you want to create quick bug" for IDE, and "If you want to scale by distributing" for Grid.

When it comes to automation, selenium is one of the popular open-source automation testing tools used in testing different web applications. It is a customizable tool available in many languages and browsers worldwide. This prime tool is used by quality analysts (junior testers to lead testers) to carry out test automation.

No matter how best a tool is, it has some pros and cons. Let's understand:

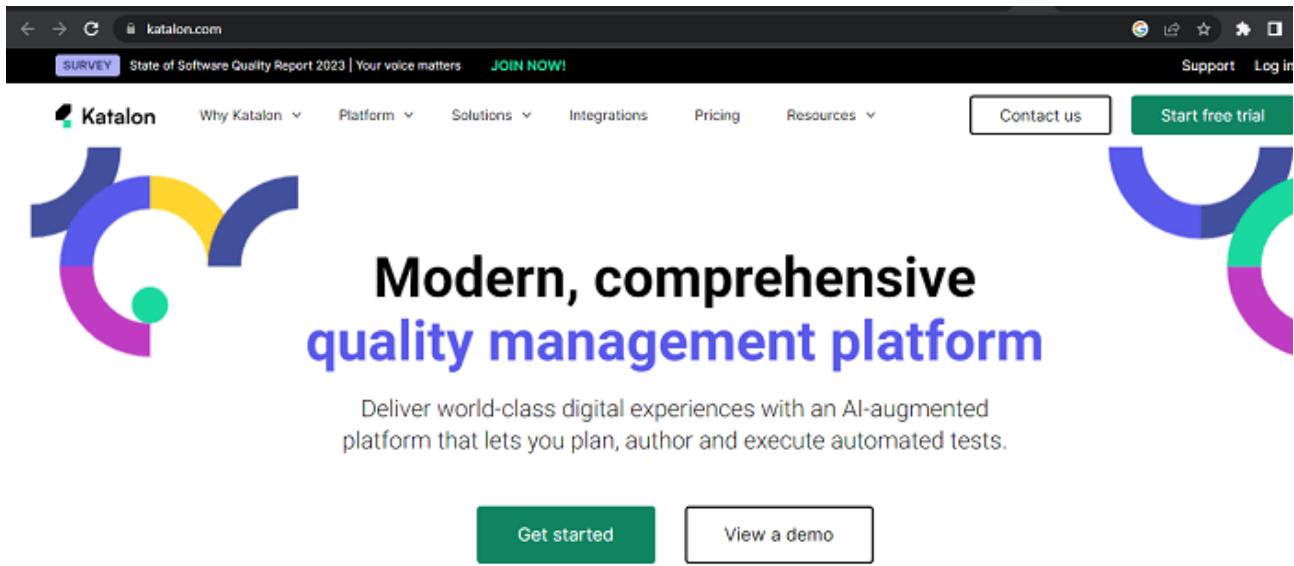
| Advantage                                                                                                                                                                                                                                                                                                                                                                                                                    | Disadvantage                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>○ It is an open-source tool freely available to everyone. Therefore, anyone can download it and use it for free.</li> <li>○ It is a flexible tool that is adopted and adapted by companies according to their needs.</li> <li>○ Selenium can support any project written in any language. For example, some projects using Java and PHP languages can use selenium to test</li> </ul> | <ul style="list-style-type: none"> <li>○ Selenium is best suited for testing web applications, but when it comes to testing native mobile apps, hybrid apps and desktop apps, selenium is not a good option.</li> <li>○ Selenium failed to provide expected tech support in testing compared to the paid tools.</li> <li>○ Testing with selenium doesn't support generating testing reports, and testers have to use frameworks</li> </ul> |

- both.
- Selenium also supports devices and platforms like iPhone, Android and Blackberry for testing mobile web applications.
  - The tester can perform parallel testing using the selenium grid, which reduces the execution time and improves productivity.
  - With programming languages, selenium also supports various operating systems (Windows, Unix, Linux and Mac) and different browsers (Firefox, Opera, Google, and Safari).

like JUnit and TestNG to generate them.

- In test management task, selenium lack built-in tools that result in a communication gap between testers and developers.
- Selenium takes a lot of time and expertise to set up due to several plugins and tools needed in automation testing that can only be done with manual configuration.

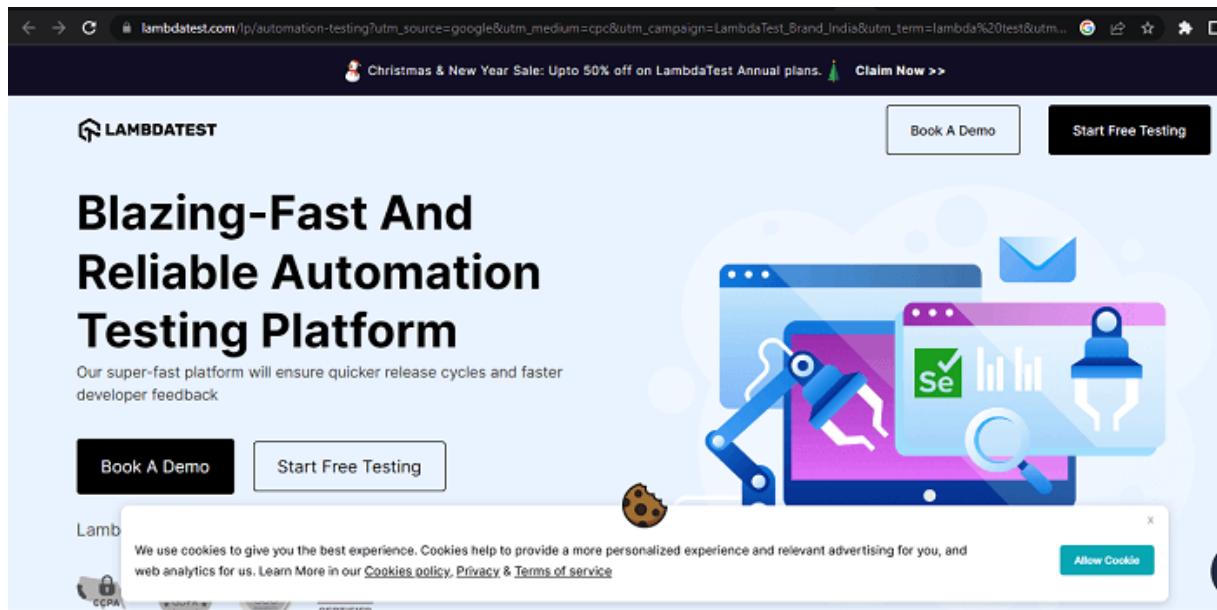
### Katalon:



The screenshot shows the Katalon website homepage. At the top, there's a navigation bar with links for SURVEY, State of Software Quality Report 2023 | Your voice matters, JOIN NOW!, Support, and Log in. Below the navigation is a large green button with white text that says "Start free trial". To the left of the button is a "Contact us" link. The main content area features the Katalon logo (a stylized 'K' composed of overlapping colored arcs in blue, yellow, purple, and green) on the left. To the right of the logo, the text "Modern, comprehensive quality management platform" is displayed in large, bold, black and blue letters. Below this text is a subtitle: "Deliver world-class digital experiences with an AI-augmented platform that lets you plan, author and execute automated tests." At the bottom of the page, there are two buttons: a green "Get started" button and a white "View a demo" button. A progress bar at the very bottom indicates a "Sending request..." process.

Katalon is a cross-browser and free licensed tool developed in 2015. It is used for automation testing of web interfaces, APIs and mobile (iOS and Android). This tool relies on the automation frameworks of Appium and Selenium.

- With Katalon, testers can perform local and remote testing and support sequential and parallel executions. One feature is a dual scripting interface that makes a non-coding skill user to use a straightforward interface.
- **Lambada test** is a cloud-based automation tool that allows testers to record real-time browser compatibility testing. It is best suited for mobile and desktop applications that allow testing (manual and automatic) of more than 2000 operating systems, devices and browsers. The Lambada test supports various frameworks and tools, like a playwright, Appium, Cypress, Tiako, Selenium, XCUITest, Espresso, Puppeteer and many more. It is a powerful tool where test execution takes place in a better way because here tester can track the issues and sort them out. It also integrates with various CD/CI tools like Travis CI, Circle CI, Jenkins, etc.

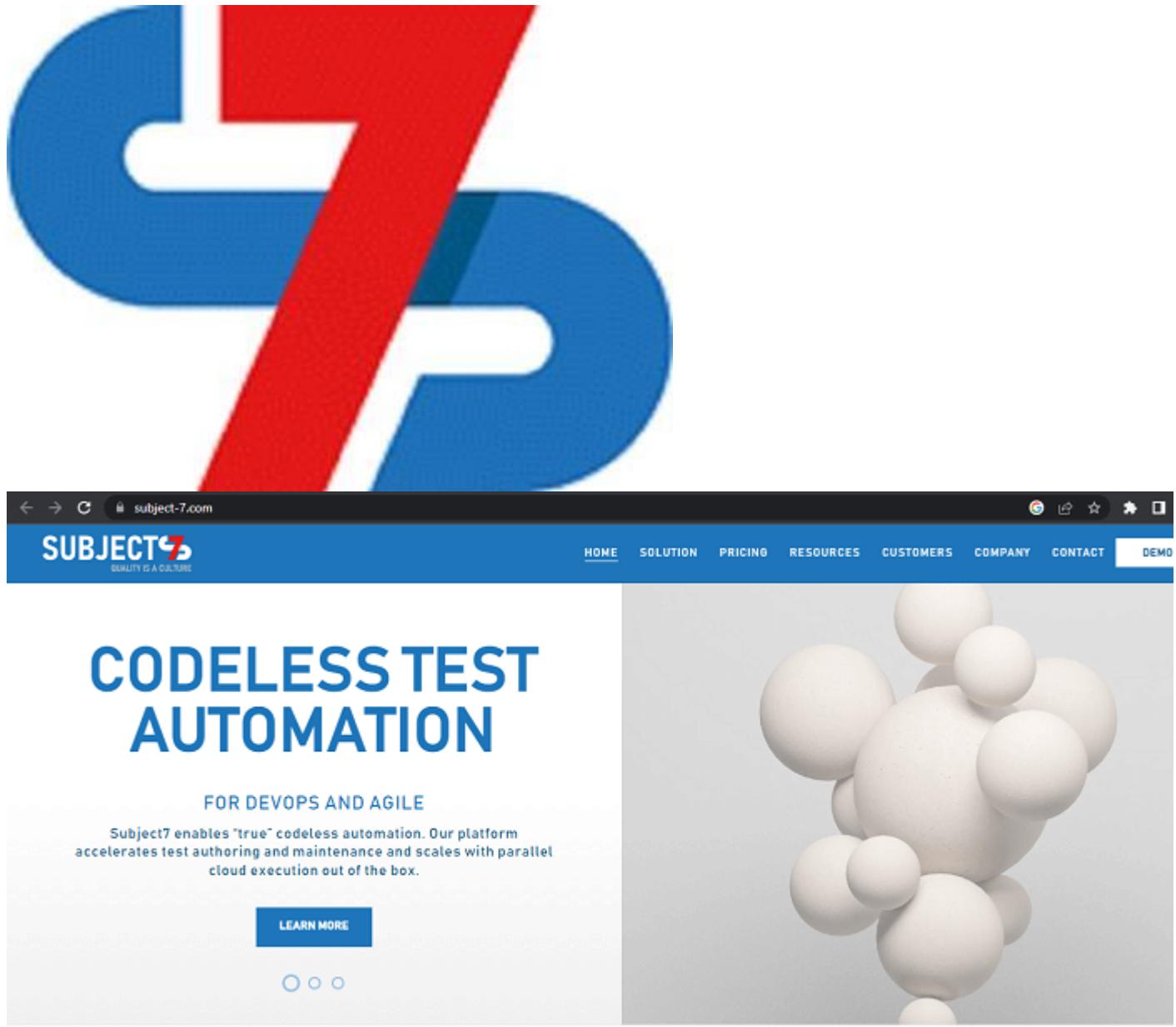


- **Test complete:** One of the top automation testing tools for web applications, desktop and mobile, in which the tester can build and run functional UI tests by replay capabilities, robust records and scripting in your languages (javascript, VB script, Python etc.). You can search for this automation testing tool on [www.testmo.com](http://www.testmo.com), developed by Smartbear software.

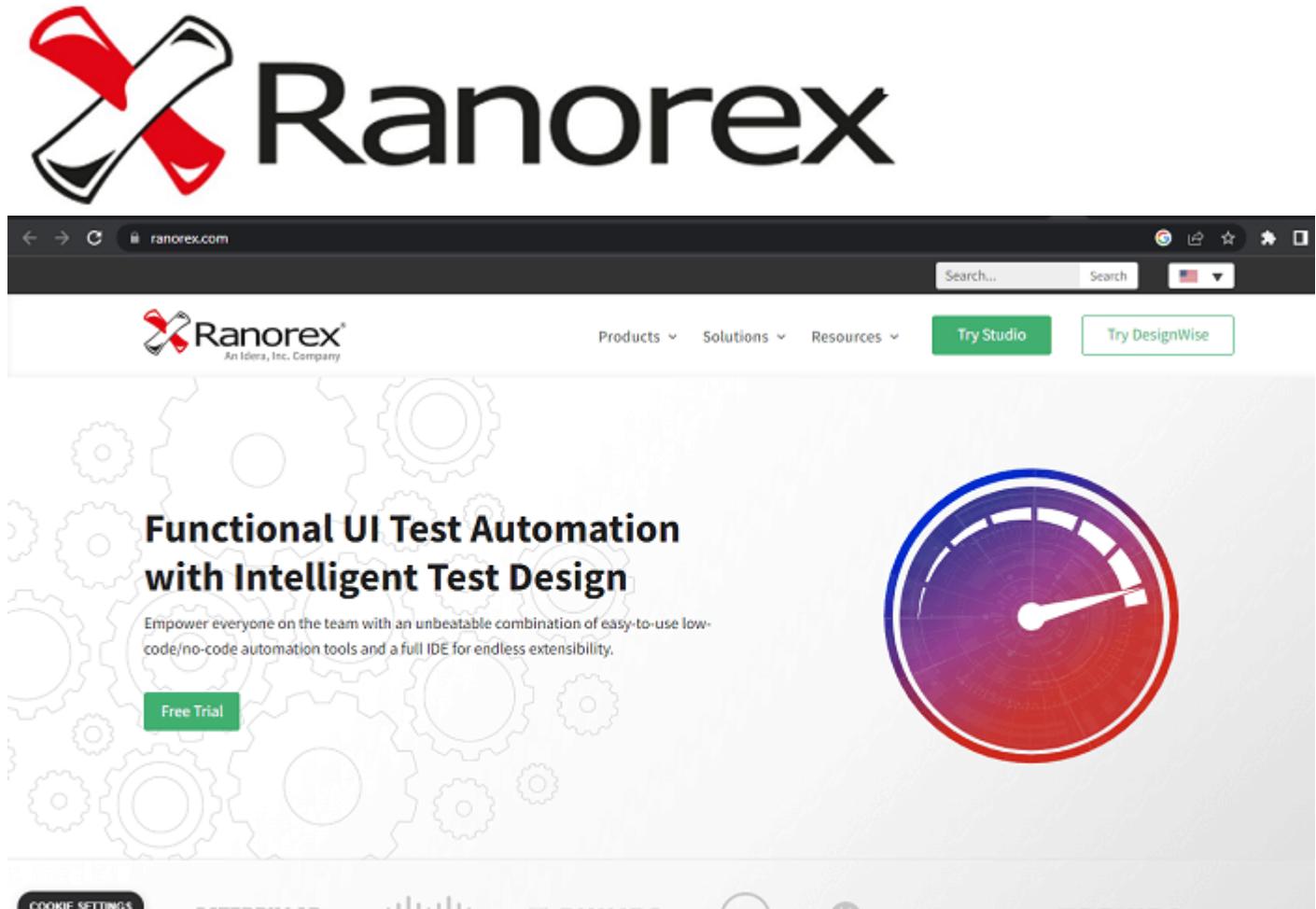


# TestComplete

- **Subject 7:** It is an all-rounder cloud-based automation testing tool that can empower anyone to become an automation expert. It is a codeless test automation solution that reduces test maintenance, accelerates test authoring and scales effortlessly. It supports end-to-end, functional, API, regression, and database testing with non-functional testing (including security, load and accessibility).
- This tool easily integrates with Agile/DevOps tooling using in-app integrations, native plugins and open APIs.



- **Ranorex:** It is one of the automation testing tools designed for mobile, desktop and web applications. It is a beneficial tool in many ways as it allows: codeless test creation, replaying testing phases, and recording and reusable test scripts. It saves testers time by offering a sharable repository and video reporting of test execution. With these features, it is best suited for beginners and experts.

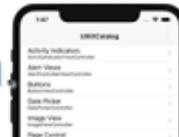


- **Appium:** An open-source test automation framework specially designed to test the mobile application (Android and iOS platforms). It is a client-based server architecture that supports multiple programming languages for writing tests like JavaScript, PHP, Java, Python etc. It provides cross-platform testing and reusability of code in testing and also can record the gestures as code.



The screenshot shows the official Appium website at [appium.io](https://appium.io). The header includes the Appium logo and navigation links for Appium, Home, Introduction, Get started, History, Get Involved!, Documentation, and Books & Resources. A sidebar on the left contains links like 'Login sequence', 'Adding and manipulating a contact', 'Adding and manipulating a dream', and 'Adding and manipulating a note'. The main content area features the Appium logo and the heading 'Automation for Apps'. Below it, a paragraph explains that Appium is an open source test automation framework for native, hybrid, and mobile web apps, driven by iOS, Android, and Windows using the WebDriver protocol. A prominent blue button labeled 'Download Appium' is centered. At the bottom, there are links for 'View on GitHub', 'Downloads', 'Examples', and 'Contributing'.

## Introducing Appium



- o **Eggplant:**

For different types of testing, Test Plant developed Eggplant, considered a suite of tools for automation testing where different types of testing are performed by each tool. Its performance testing tool (Eggplant Performance) handles the stress testing, performance and load, and the functional testing tool looks what the name indicates.

Other automation testing tools work on an object-based approach, but Eggplant follows an image-based approach. In this testing tool, testers interact with the applications as same as end users will interact. Eggplant Digital Automation Intelligence is used by the tester, particularly for GUI and application testing.



Mitigate Risk & Release Confidently

Take our 2-minute assessment and get a customized report to identify your risk factors and where you are exposed, and how test automation can help.

Take Risk Assessment

Your Technology Choices Matter. Get the 2021 Market Guide for AI-Augmented Software Testing Tools from Gartner®.

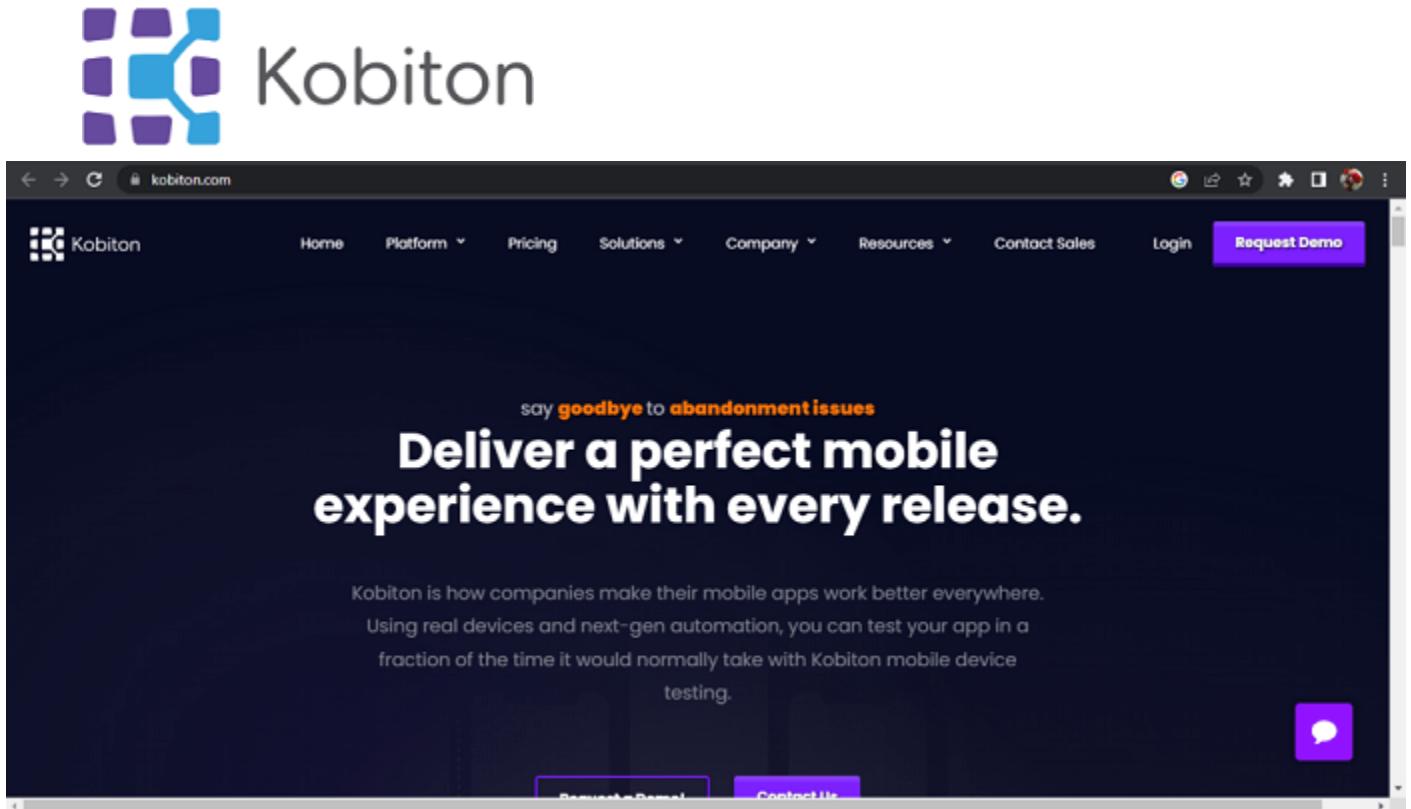
- **Cucumber:** Technically, the cucumber automation tool is the right choice for quality analyst testers that focus on the end-user experience above all the factors. Therefore, it is called BBD(behavioural-driven development tool). It is an open-source automation tool that supports various languages such as Scala, Java, groovy, Ruby etc.

In cucumber, a tester can get end-to-end testing framework support where test code is written in a simple language (English) known as Gherkin, which can be executed on a different framework. It is the best tool from a business perspective as it supports behaviour-driven development in which business analysts and product owners write test scenarios to enact the system behaviour.

The screenshot shows the official Cucumber website at cucumber.io. The header features the Cucumber logo (a stylized green flower) and navigation links for Tools, Docs, Learn BDD, and Resources. A search bar and login button are also present. A banner at the top right encourages users to "Try CucumberStudio". A prominent call-to-action button says "CukenFest 2023 tickets are now available - get yours here!". The main content area has a dark background with two people in glasses looking at a screen. On the left, there's a section titled "Tools & techniques that elevate teams to greatness" with a subtext about BDD. On the right, a code editor window displays a portion of a Cucumber feature file:

```
Comment
@tag
Feature: Eating too many cucumbers may not be good for you
 Eating too much of anything may not be good for you.
 Scenario: Eating a few is no problem
 Given Alice is hungry
 When she eats 2 cucumbers
 Then she is full
```

- **Kobiton:** Kobiton is a wonderful cloud-based platform automation testing tool for mobile and IoT (internet of things) continuous testing (manual and automated). Functionality, compatibility, visual and UX and performance testing can be automated using the AI-driven scriptless approach of Kobiton. This tool supports Katalon Studio, Selenium Web Driver and Appium with CI/CD integrations like Jenkins, TeamCity, GitHub etc. Kobiton can capture screenshots and user interactions via recording videos. It is the fastest processing mobile testing tool that keeps up with DevOps' speed.



- **Telerik Test Studio:** Telerik has launched this latest automation tool named Test Studio which is a record and playback tool and supports all these:

→ cross-browser

→ Scripting language VB.Net and C#

→ Automating applications such as JavaScript, MVC, Ruby, WPF, HTML5, PHP, Android, Angular, Silverlight, iOS and PHP.

Tests can be scheduled with this tool, and further can be executed in parallel, and reports are also provided by it.

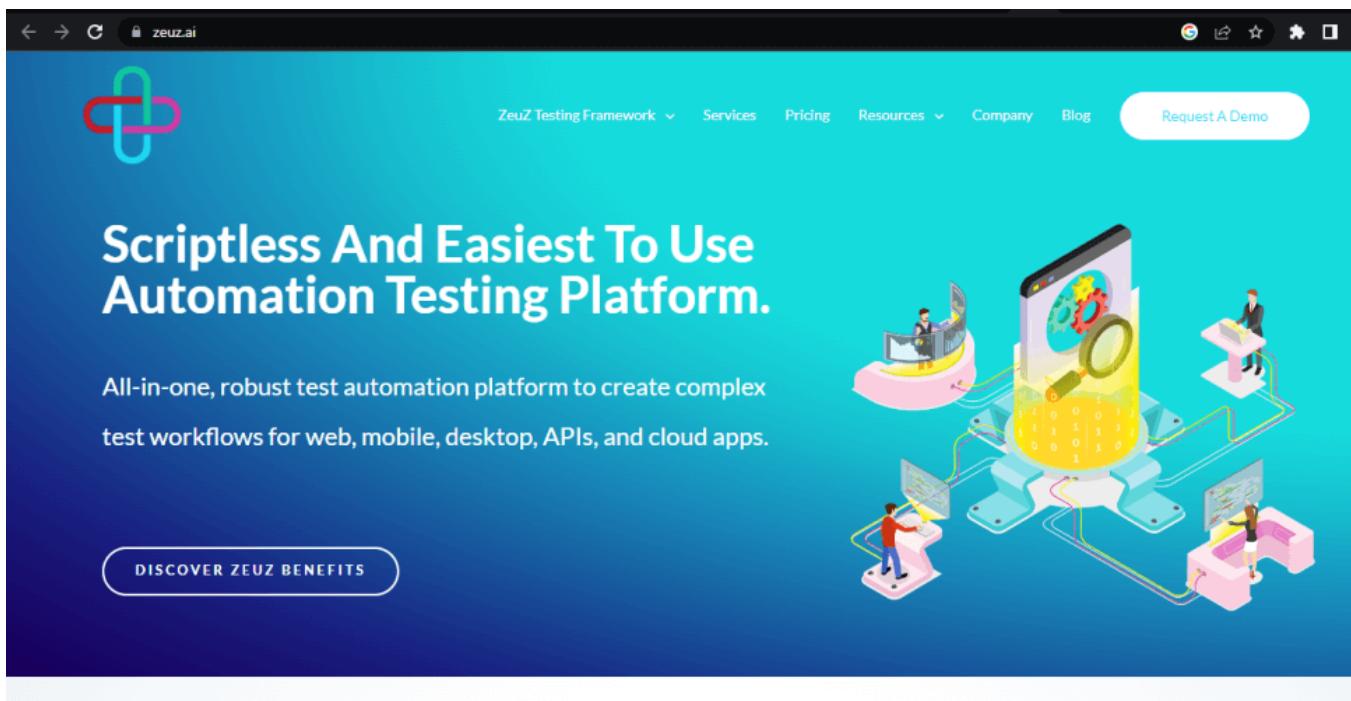
For better performance and results, it connects with source control systems such as GIT and Team Foundation Server and further does continuous testing.

- **Cypress:** Like cucumber, this automation tool focuses on end-to-end testing with modern JavaScript frameworks. It has various features like it provides in-depth and detailed documentation, having an inbuilt large number of libraries, DOM manipulation and shadow DOM. Powerful end-to-end testing scenarios with lightning fast test creation and execution can be created with DOM. It has some advantages and disadvantages too.

| Advantage                                                                                                                                                                                    | Disadvantage                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>○ Fastest test creation and execution</li> <li>○ Specializing in end-to-end testing</li> <li>○ Excellent detail and in-depth documentation</li> </ul> | <ul style="list-style-type: none"> <li>○ Learning curve</li> <li>○ Lack of support for iframes</li> <li>○ Works with browsers like Chrome and other Chromium-based browsers</li> </ul> |

- **ZeuZ:** This tool is one of the simple, robust, scriptless and AI-assisted click and test automation framework tools that provide to-end automation. It works on cross platforms such as API, mobile, cloud services, web, desktop and IoT. It is the complete package as it does regression, performance, and management and run manual +automated, data-driven tests, UI and functional testing.

ZeuZ is equipped with intelligent debugging, collaboration features, rich reporting, CI/CD integration, notifications and batch updates. It does recording and playback all the test cases. It does advanced debugging and flexible deployment. ZeuZ has a built-in waiting mechanism and AI-powered object identification. It's a trail automation test available for free.



- **Tenjin Online:** A 5<sup>th</sup>-generation testing tool that supports manual and automation test execution in which having technical knowledge is not mandatory as it is a codeless tool. It provides SaaS-based solutions to test efficiently. It can test applications from anywhere, anytime, because it doesn't require critical configuration or skills that also make easy execution of functional and UI tests. It allows cross-browser testing and an efficient and quick independent test platform. All the Tenjin online tool features reduce testing time and cost by upto 75%.

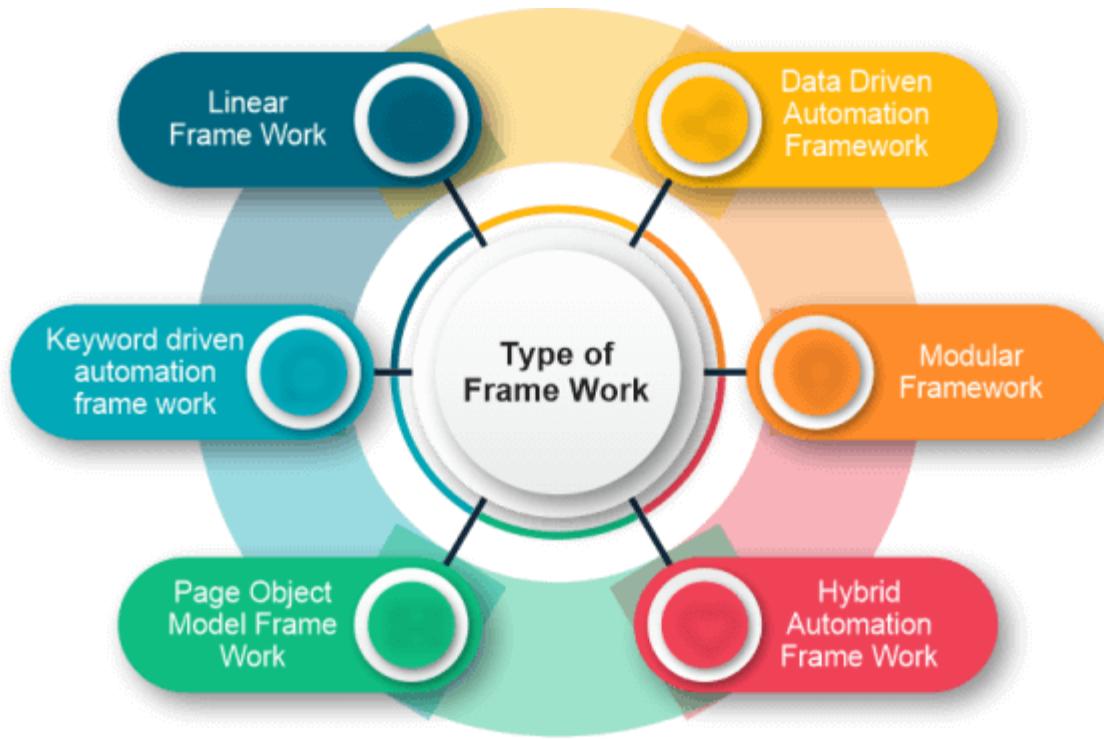
The screenshot shows the homepage of Tenjin Online. At the top, there's a navigation bar with links for FEATURES, RESOURCES, PRICING, VIDEOS, ABOUT US, and CONTACT, along with a Sign-Up button. The main headline reads "End to End Automation testing for Dev/QA Teams (Web & Mobile)". Below the headline is a bulleted list of advantages: "No complex setups", "Self-assisted", "Automated testing for web and mobile apps", "Create tests 6X faster", "Extend the platform effortlessly with add-ons", and "Zero test maintenance". To the right of the text is a cartoon illustration of a person sitting at a desk with a computer, looking relaxed with their hands behind their head. There's also a small potted plant on the desk.

### Advantages of automation testing

1. It saves time and cost in testing and provides an increment in the efficiency of testing.
2. Automation testing improves the accuracy of testing
3. With automation, more cycles can be achieved
4. It also ensures consistency in testing
5. Its test scripts can be reusable
6. Ability to cover the test application features widely
7. Automation testing results are reliable
8. In this testing, human intervention is not required
9. Speedily executes the testing process frequently and thoroughly

### The framework used in automation software testing

Automation testing is executed on a few frameworks:



- **Linear Framework**

One of the simplest frameworks that act as a record and playback model. In this tester write the simple code to run the test cases without sequential steps and modularity

- **Data-driven automation framework**

The data-driven automation framework can perform both negative and positive test cases. All the test case data inputs are stored in the extension files and tables from where values are read during the execution of test scripts.

- **Modular automation framework**

A modular automation framework is best suited to run large test scripts as it divides the test scripts into independent modules that hierarchically interact with each other. These small independent modules are tested easily because it's easy to create required test scenarios.

- **Hybrid automation framework**

Hybrid is always a combination; here, keyword-driven and data-driven frameworks are combined in which test data and keywords are externalized. Test data is stored in an excel or properties file, whereas keywords are maintained in a separate java file.

- **Page object model framework**

In this framework, the tester doesn't need to write the code repeatedly because an object for the UI element is created that can be recalled later for testing. This feature of the POM framework results in less code usability and verbosity and reduces time consumption in writing test scripts.

- **Keyword-driven automation framework**

In KDF, the keywords are separated for a common set of functions and instructions, due to which automation speeds up. In this scripting technique, keywords are associated with actions like closing and opening a browser, mouse-click events, and others. During testing, all the keywords are recalled to perform specific steps, and these keywords are maintained in a file along with the actions they perform.

### Some myths about automation testing

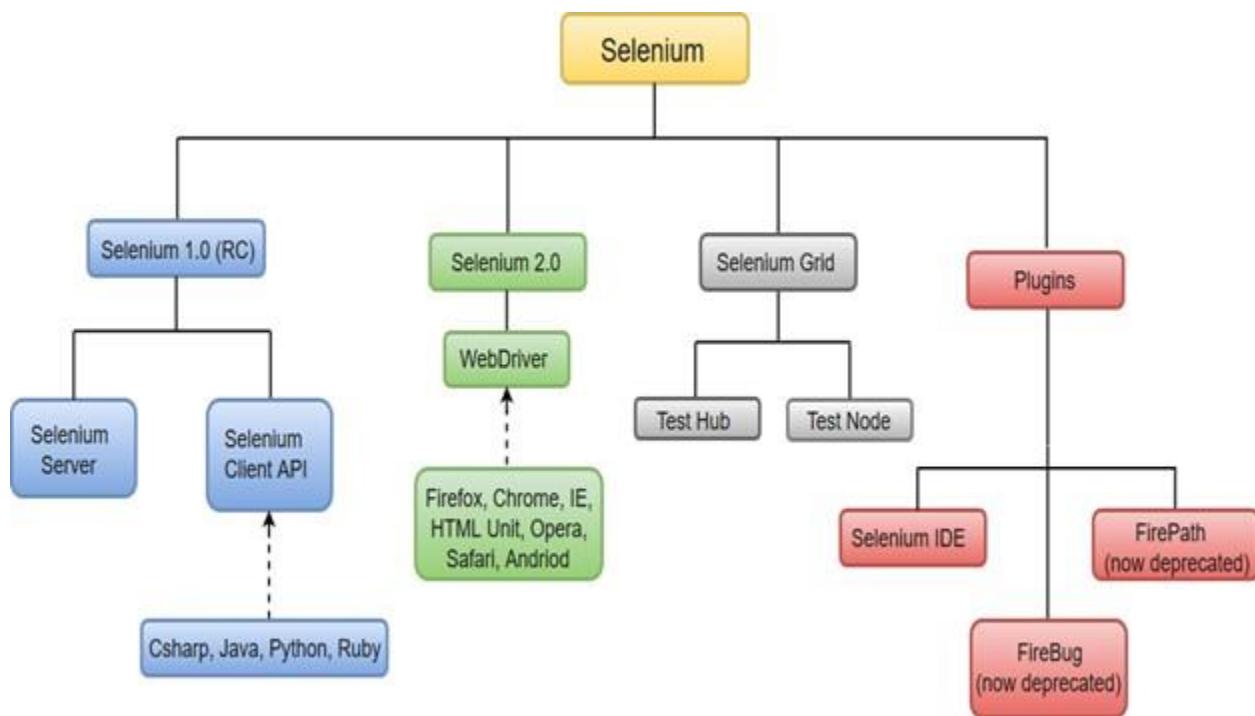
- **Automation testing is expensive-** It's a myth because it is developed to reduce time, testing effort and resource requirements. If the automation process is followed correctly will save the project cost in the long run.
- **It is possible to do 100% automation-** We have learnt that some of the test cases' exploratory and usability can't be automated, which stated that 100% automation is impossible.
- **It will finish manual testing jobs-** It's humans who made things automatic that ease the use of tools. But this ease in automation testing decreases the need for only manual testers; now, the company needs full-stack or dual-role testers capable of working on both automation and manual testing.
- **A developer can be a good automation tester** -Rather, a developer develops the whole software but doesn't have a tester perspective for testing the software. Therefore, a developer has command of coding, but a tester can think beyond testing.
- **Anyone can do automation testing** - Today, various automation tools are freely available on the internet, and the testing procedure is too. Still, a normal software user couldn't understand the technicality and carry out the testing. To do this, one must know about testing.

### Automate Testing of Web Applications

## Selenium: Introducing Web Driver and Web Elements

Selenium WebDriver is the most important component of Selenium Tool's Suite. The latest release "Selenium 2.0" is integrated with WebDriver API which provides a simpler and more concise programming interface.

The following image will give you a fair understanding of Selenium components and the Test Automation Tools.



Selenium WebDriver was first introduced as a part of Selenium v2.0. The initial version of Selenium i.e Selenium v1 consisted of only IDE, RC and Grid. However, with the release of Selenium v3, RC has been deprecated and moved to legacy package.

In WebDriver, test scripts can be developed using any of the supported programming languages and can be run directly in most modern web browsers. Languages supported by WebDriver include C#, Java, Perl, PHP, Python and Ruby.

Before learning the concepts of Selenium WebDriver, you should be well versed with any of the supported programming languages. Currently, Selenium Web driver is most popular with Java and C#. For this tutorial, we are using Selenium with java. You can refer to the links given below to learn basic as well as advance concepts of Java and C#:

Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the browser.

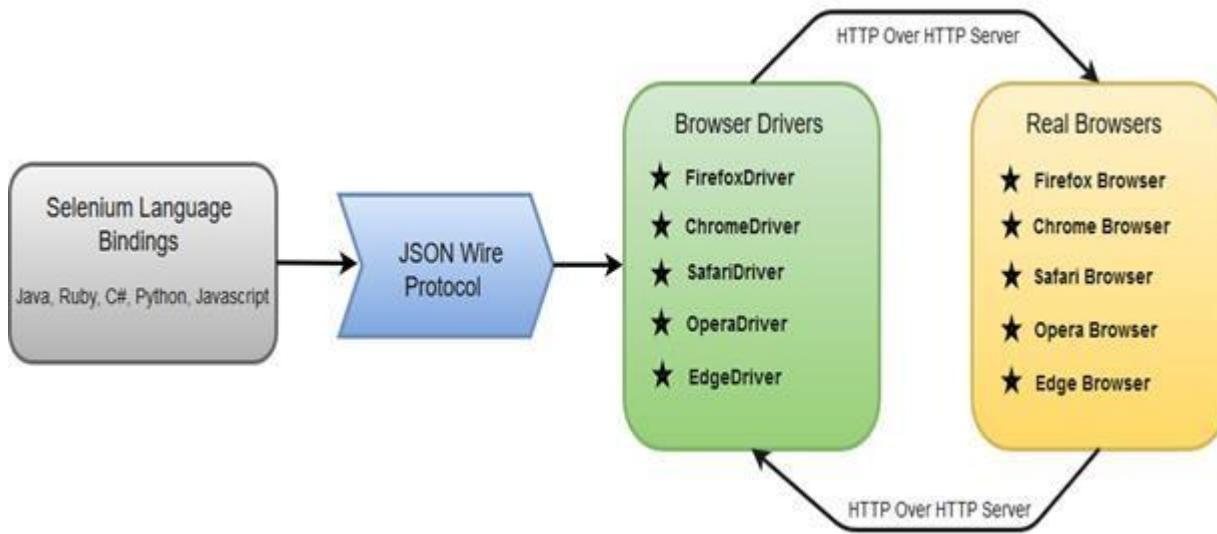
WebDriver has a built-in implementation of Firefox driver (Gecko Driver). For other browsers, you need to plug-in their browser specific drivers to communicate and run the test. Most commonly used WebDriver's include:

- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- Safari Driver
- HTML Unit Driver (a special headless driver)

### Selenium WebDriver- Architecture

Selenium WebDriver API provides communication facility between languages and browsers.

The following image shows the architectural representation of Selenium WebDriver.



There are four basic components of WebDriver Architecture:

- Selenium Language Bindings
- JSON Wire Protocol
- Browser Drivers
- Real Browsers

### Selenium Language Bindings / Selenium Client Libraries

Selenium developers have built language bindings/Selenium Client Libraries in order to support multiple languages. For instance, if you want to use the browser driver in java, use the java bindings. All the supported language bindings can be downloaded from the official website (<https://www.seleniumhq.org/download/#client-drivers>) of Selenium.

### JSON Wire Protocol

JSON (JavaScript Object Notation) is an open standard for exchanging data on web. It supports data structures like object and array. So, it is easy to write and read data from JSON. To learn more about JSON, visit <https://www.javatpoint.com/json-tutorial>

JSON Wire Protocol provides a transport mechanism to transfer data between a server and a client. JSON Wire Protocol serves as an industry standard for various REST web services. To learn more about Web Services, visit <https://www.javatpoint.com/web-services-tutorial>

## Browser Drivers

Selenium uses drivers, specific to each browser in order to establish a secure connection with the browser without revealing the internal logic of browser's functionality. The browser driver is also specific to the language used for automation such as Java, C#, etc.

When we execute a test script using WebDriver, the following operations are performed internally.

- HTTP request is generated and sent to the browser driver for each Selenium command.
- The driver receives the HTTP request through HTTP server.
- HTTP Server decides all the steps to perform instructions which are executed on browser.
- Execution status is sent back to HTTP Server which is subsequently sent back to automation script.

## Browsers

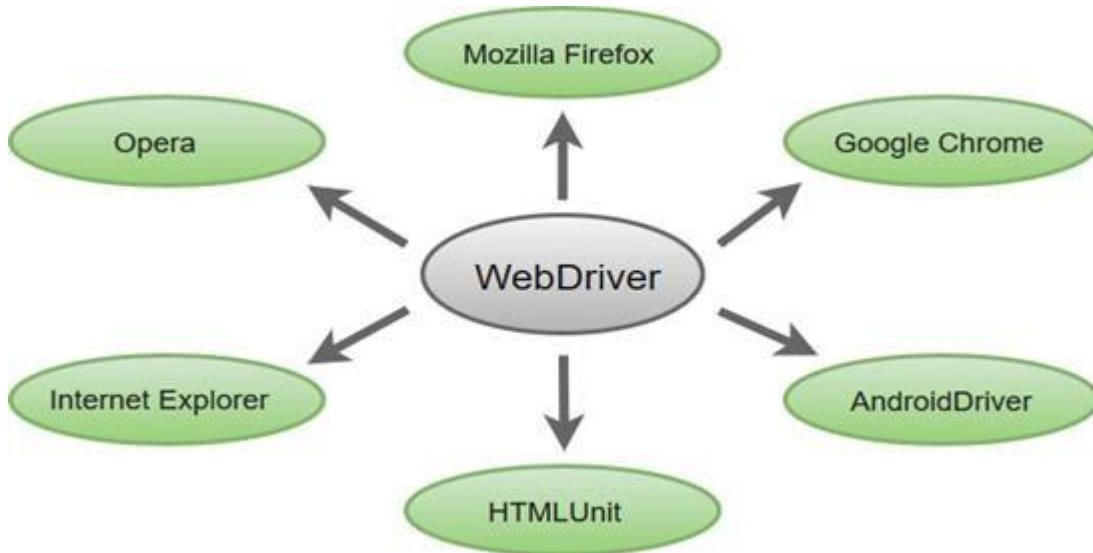
Browsers supported by Selenium WebDriver:

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Safari

## Selenium WebDriver- Features

Some of the most important features of Selenium WebDriver are:

- **Multiple Browser Support:** Selenium WebDriver supports a diverse range of web browsers such as Firefox, Chrome, Internet Explorer, Opera and many more. It also supports some of the non-conventional or rare browsers like HTMLUnit.



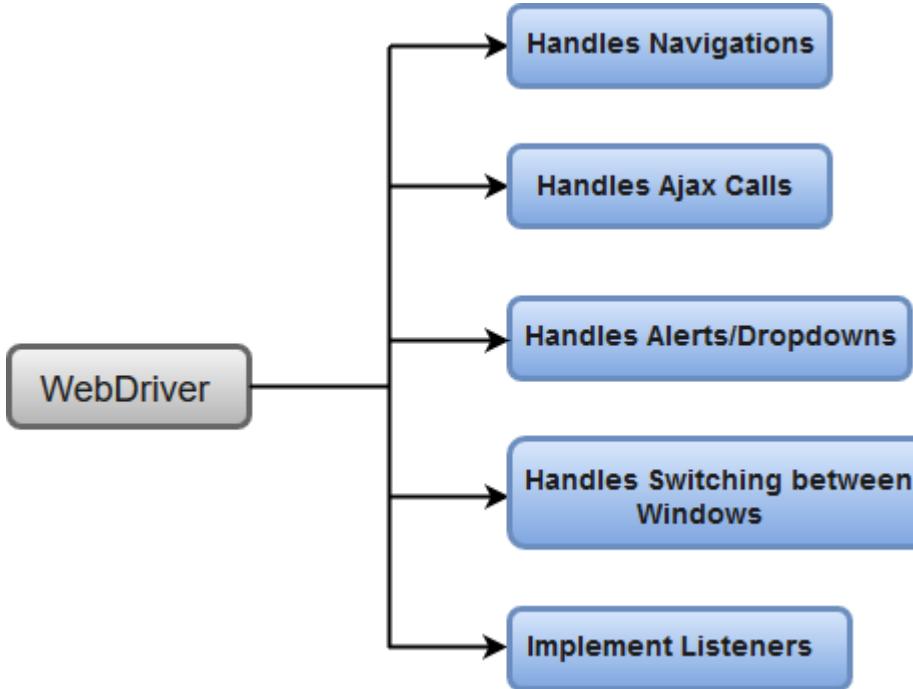
- **Multiple Languages Support:** WebDriver also supports most of the commonly used programming languages like Java, C#, JavaScript, PHP, Ruby, Pearl and Python. Thus, the user can choose any one of the supported programming language based on his/her competency and start building the test scripts.
- **Speed:** WebDriver performs faster as compared to other tools of Selenium Suite. Unlike RC, it doesn't require any intermediate server to communicate with the browser; rather the tool directly communicates with the browser.



- **Simple Commands:** Most of the commands used in Selenium WebDriver are easy to implement. For instance, to launch a browser in WebDriver following commands are used:
 

```
WebDriver driver = new FirefoxDriver(); (Firefox browser)
WebDriver driver = new ChromeDriver(); (Chrome browser)
WebDriver driver = new InternetExplorerDriver(); (Internet Explorer browser)
```
- **WebDriver- Methods and Classes:** WebDriver provides multiple solutions to cope with some potential challenges in automation testing.

WebDriver also allows testers to deal with complex types of web elements such as checkboxes, dropdowns and alerts through dynamic finders.



---

### Selenium WebDriver Tutorial Index

---

#### Selenium WebDriver Tutorial

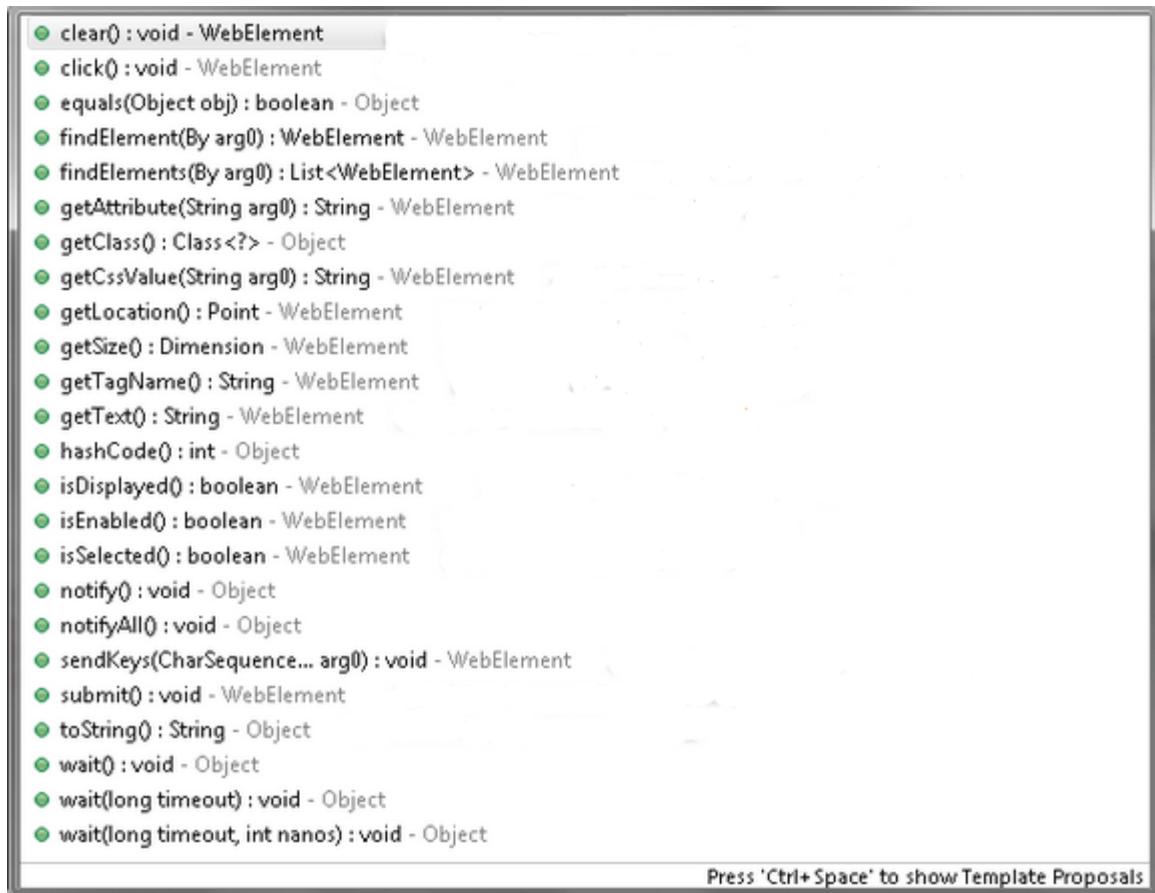
- [Selenium WebDriver Tutorial](#)
- [WebDriver Vs RC](#)
- [WebDriver-Installation](#)
- [WebDriver First Test Case](#)
- [WebDriver Commands](#)
- [Running Test on Chrome](#)
- [Running Test on Firefox](#)
- [Running Test on IE](#)
- [Running Test on Safari](#)

- Locating Strategies
- Locating Strategies By ID
- Locating Strategies By Name
- Locating Strategies By Class Name
- Locating Strategies By Tag Name
- Locating Strategies By Link Text
- Locating Strategies By Partial Link Text
- Locating Strategies By CSS
- Locating Strategies By XPath
- Handling Drop-Downs
  
- WebDriver-Drag and Drop
- WebDriver-Handling Alerts
- Scrolling a Web Page
- WebDriver - Browser Commands
- WebDriver - Navigation Commands
- WebDriver - WebElement Commands
- Handling Radio Buttons
- Handling Checkbox
- Selenium Assertions
- Selenium Grid

### What is Web Element?

The term web element refers to a HTML element. The HTML documents are composed of HTML elements. It consists **a start tag, an end tag** and the **content** in between. For instance, a HTML element is written as: "<tagname> content </tagname>"

In WebDriver, we have several commonly used web element commands and actions. The following screenshot displays the eclipse web element command panel.



1. `WebElement element = driver.findElement(By.id("UserName"));`

Here, the `UserName` is the value of the `id` attribute, used as a unique identification for the desired web element.

Given are some of the most commonly used `WebElement` commands for Selenium WebDriver.

### 1. Clear Command

#### Method:

1. `clear() : void`

#### Command:

1. `element.clear();`

#### Code snippet:

```
1. WebElement element = driver.findElement(By.id("UserName"));
2. element.clear();
3.
4. //Or can be written as
5.
6. driver.findElement(By.id("UserName")).clear();
```

## 2. Sendkeys Command

### Method:

```
1. sendKeys(CharSequence? KeysToSend) : void
```

### Command:

```
1. element.sendKeys("text");
```

### Code snippet:

```
1. WebElement element = driver.findElement(By.id("UserName"));
2. element.sendKeys("JavaTpoint");
3.
4. //Or can be written as
5.
6. driver.findElement(By.id("UserName")).sendKeys("JavaTpoint");
```

## 3. Click Command

### Method:

```
1. click() : void
```

### Command:

```
1. element.click();
```

### Code snippet:

```
1. WebElement element = driver.findElement(By.linkText("javaTpoint"));
```

```
2. element.click();
3.
4. //Or can be written as
5.
6. driver.findElement(By.linkText("javaTpoint")).click();
```

#### 4. IsDisplayed Command

##### Method:

```
1. isDisplayed() : boolean
```

##### Command:

```
1. element.isDisplayed();
```

##### Code snippet:

```
1. WebElement element = driver.findElement(By.id("UserName"));
2. boolean status = element.isDisplayed();
3.
4. //Or can be written as
5.
6. boolean status = driver.findElement(By.id("UserName")).isDisplayed();
```

#### 5. IsEnabled Command

##### Method:

```
1. isEnabled() : boolean
```

##### Command:

```
1. element.isEnabled();
```

##### Code snippet:

```
1. WebElement element = driver.findElement(By.id("UserName"));
2. boolean status = element.isEnabled();
```

```
3.
4. //Or can be written as
5.
6. boolean staus = driver.findElement(By.id("UserName")).isEnabled();
7.
8. //Or can be used as
9. WebElement element = driver.findElement(By.id("userNaMe"));
10. boolean status = element.isEnabled();
11. // Check that if the Text field is enabled, if yes enter value
12. if(status){
13. element.sendKeys("javaTpoinT");
14. }
```

## 6. IsSelected Command

### Method:

```
1. isSelected() : boolean
```

### Command:

```
1. element.isSelected();
```

### Code snippet:

```
1. WebElement element = driver.findElement(By.id("Sex-Male"));
2. boolean status = element.isSelected();
3.
4. //Or can be written as
5.
6. boolean staus = driver.findElement(By.id("Sex-Male")).isSelected();
```

## 7. Submit Command

### Method:

```
1. submit() : void
```

**Command:**

1. element.submit();

**Code snippet:**

```
1. WebElement element = driver.findElement(By.id("SubmitButton"));
2. element.submit();
3.
4. //Or can be written as
5.
6. driver.findElement(By.id("SubmitButton")).submit();
```

**8. GetText Command****Method:**

1. getText() : String

**Command:**

1. element.getText();

**Code snippet:**

```
1. WebElement element = driver.findElement(By.xpath("anyLink"));
2. String linkText = element.getText();
```

**9. GetTagName Command****Method:**

1. getTagName() : String

**Command:**

1. element.getTagName();

**Code snippet:**

```
1. WebElement element = driver.findElement(By.id("SubmitButton"));
```

```
2. String tagName = element.getTagName();
3.
4. //Or can be written as
5.
6. String tagName = driver.findElement(By.id("SubmitButton")).getTagName();
```

## 10. getCssValue Command

### Method:

```
1. getCssvalue() : String
```

### Command:

```
1. element.getCssValue();
```

## 11.getAttribute Command

### Method:

```
1. getAttribute(String Name) : String
```

### Command:

```
1. element.getAttribute();
```

### Code snippet:

```
1. WebElement element = driver.findElement(By.id("SubmitButton"));
2. String attValue = element.getAttribute("id"); //This will return "SubmitButton"
```

## 12. getSize Command

### Method:

```
1. getSize() : Dimension
```

### Command:

```
1. element.getSize();
```

**Code snippet:**

```
1. WebElement element = driver.findElement(By.id("SubmitButton"));
2. Dimension dimensions = element.getSize();
3. System.out.println("Height :" + dimensions.height + "Width :" + dimensions.width)
;
;
```

**13. getLocation Command****Method:**

```
1. getLocation() : Point
```

**Command:**

```
1. element.getLocation();
```

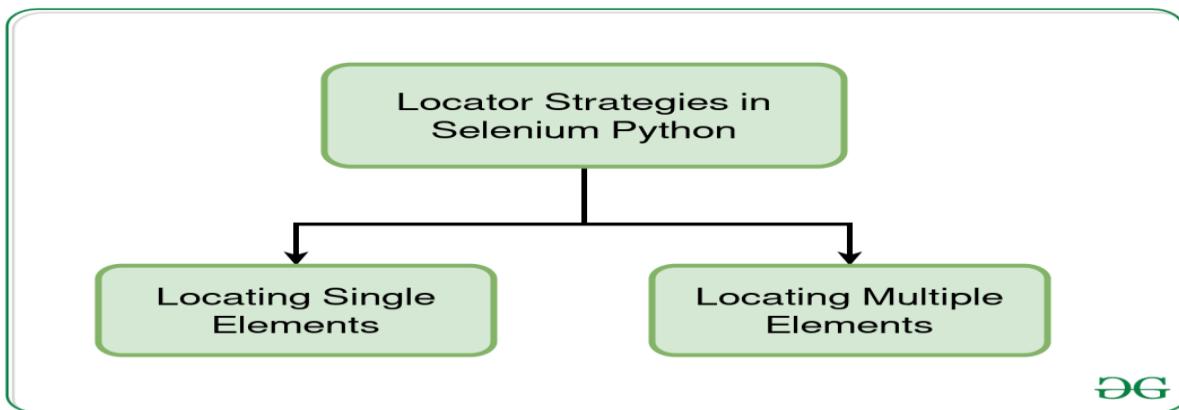
**Code snippet:**

```
1. WebElement element = driver.findElement(By.id("SubmitButton"));
2. Point point = element.getLocation();
3. System.out.println("X coordinate : " + point.x + "Y coordinate: " + point.y);
```

**Locating Web Elements**

Locators Strategies in Selenium Python are methods that are used to locate elements from the page and perform an operation on the same. Selenium's Python Module is built to perform automated testing with Python. Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. After one has [installed selenium](#) and checked out – [Navigating links using get method](#), one might want to play more with Selenium Python. After opening a page using selenium such as geeksforgeeks, one might want to click some buttons automatically or fill a form automatically or any such automated task. This article revolves around two strategies – Locating Single Elements and Location Multiple

Elements.



### **Locator Strategies to locate single first elements**

Selenium Python follows different locating strategies for elements. One can locate a element in 8 different ways. Here is a list of locating strategies for Selenium in python

| Locators                                                 | Description                                                                                |
|----------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <a href="#"><u>find_element_by_id</u></a>                | The first element with the id attribute value matching the location will be returned.      |
| <a href="#"><u>find_element_by_name</u></a>              | The first element with the name attribute value matching the location will be returned.    |
| <a href="#"><u>find_element_by_xpath</u></a>             | The first element with the xpath syntax matching the location will be returned.            |
| <a href="#"><u>find_element_by_link_text</u></a>         | The first element with the link text value matching the location will be returned.         |
| <a href="#"><u>find_element_by_partial_link_text</u></a> | The first element with the partial link text value matching the location will be returned. |
| <a href="#"><u>find_element_by_tag_name</u></a>          | The first element with the given tag name will be returned.                                |

| Locators                                            | Description                                                                |
|-----------------------------------------------------|----------------------------------------------------------------------------|
| <a href="#"><u>find_element_by_class_name</u></a>   | the first element with the matching class attribute name will be returned. |
| <a href="#"><u>find_element_by_css_selector</u></a> | The first element with the matching CSS selector will be returned.         |

### Locator Strategies to locate multiple elements

Selenium Python follows different locating strategies for elements. One can locate multiple elements in 7 different ways. Here is a list of locating strategies for Selenium in python –

| Locators                                                  | Description                                                                       |
|-----------------------------------------------------------|-----------------------------------------------------------------------------------|
| <a href="#"><u>find_elements_by_name</u></a>              | All elements with name attribute value matching the location will be returned.    |
| <a href="#"><u>find_elements_by_xpath</u></a>             | All elements with xpath syntax matching the location will be returned.            |
| <a href="#"><u>find_elements_by_link_text</u></a>         | All elements with link text value matching the location will be returned.         |
| <a href="#"><u>find_elements_by_partial_link_text</u></a> | All elements with partial link text value matching the location will be returned. |
| <a href="#"><u>find_elements_by_tag_name</u></a>          | All elements with given tag name will be returned.                                |
| <a href="#"><u>find_elements_by_class_name</u></a>        | All elements with matching class attribute name will be returned.                 |
| <a href="#"><u>find_elements_by_css_selector</u></a>      | All elements with matching CSS selector will be returned.                         |

### Actions on Web Elements

Selenium is an open-source web automation tool that supports many user actions to perform in the web browser. Automating a web page that has file upload or other functionality which requires selecting multiple elements, to perform the multiple

select actions the selenium provides a class called Actions. The Action class provides the method for Keyboard actions. The actions provided by this class are performed by an API called Advanced user interaction in selenium webdriver.

### How to Select Multiple Elements?

Usually, we click the control(ctrl) button and select the multiple elements, Actions class also provides the same kind of approach to selecting the multiple elements. The Actions class provides the Keyboards actions that are used to Click and down the control key and click the multiple elements after that the control(ctrl) will be down. To perform this we are using the keyboard actions and the Keys in the selenium.

```
keyDown(Keys.CONTROL)
.click(element1)
.click(element2)
.build();
```

This is used to select the multiple elements in the webpage, for that we need to use the Actions class,

```
Actions action=new Actions(driver);
```

After creating an object for the actions class we need to use the Action method for the series of actions to be performed.

```
Action seriesOfActions = (Action) action.keyDown(Keys.CONTROL)
.click(element1)
.click(element2)
.build();
```

Now this series of actions is performed by calling the perform() method.

```
seriesOfActions.perform();
```

### Example:

In this example program, we are navigating to the website and trying to select multiple items.

- Java

```
public class Geeks {

 public void actios() throws InterruptedException {

 ChromeDriver driver = new ChromeDriver();

 driver.manage().window().maximize();

 driver.get("https://jqueryui.com/selectable/");

 WebElement
 iframe=driver.findElement(By.tagName("iframe"));

 driver.switchTo().frame(iframe);

 WebElement
 element1=driver.findElement(By.xpath("//li[contains(text(),'Item
1')]"));

 WebElement
 element2=driver.findElement(By.xpath("//li[contains(text(),'Item
2')]"));

 Actions action=new Actions(driver);

 Action seriesOfActions = (Action)
action.keyDown(Keys.CONTROL)

 .click(element1)

 .click(element2)

.build();
```

```
seriesOfActions.perform();

Thread.sleep(3000);

driver.close();

}

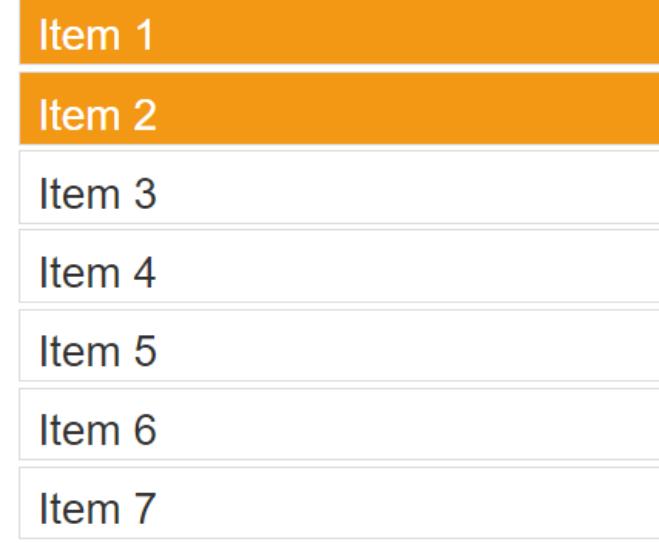
}
```

This code will navigate to the webpage and select the first two elements. This is performed by “**keyDown(Keys. CONTROL)**” it is used to click and hold the control(ctrl) key and then click operations are performed. The output of this code is,

#### Output:

---

*Use the mouse to select elements, individually or in a group.*



|        |
|--------|
| Item 1 |
| Item 2 |
| Item 3 |
| Item 4 |
| Item 5 |
| Item 6 |
| Item 7 |

## Testing: Understanding Testing.xml, Adding Classes, Packages, Methods to Test, Test Reports

TestNG, you can define multiple test cases in a single class whereas, in Java, you can define only one test in a single class in the main() method. In Java, if you want to create one more test, then you need to create another java file and define the test in the main() method.

Instead of creating test cases in different classes, we recommend you to use TestNG framework that allows you to create multiple test cases in a single class.

You can create multiple test cases with the help of **@Test** annotation.

**Let's understand through an example.**

```
1. public class test
2. {
3. @Test
4. public void test1() // First test case.
5. {
6. System.out.println("test1");
7. }
8. @Test
9. public void test2() // Second test case.
10. {
11. System.out.println("test2");
12. }
```

The above code consists of a class test. The class test consists of two test cases, i.e., test1() and test2(). You can differentiate the test cases by considering the sequence of test cases. In the above code, the test case **test2()** is written in the second @Test annotation, so this test case will be considered as the second case.

**Source code**

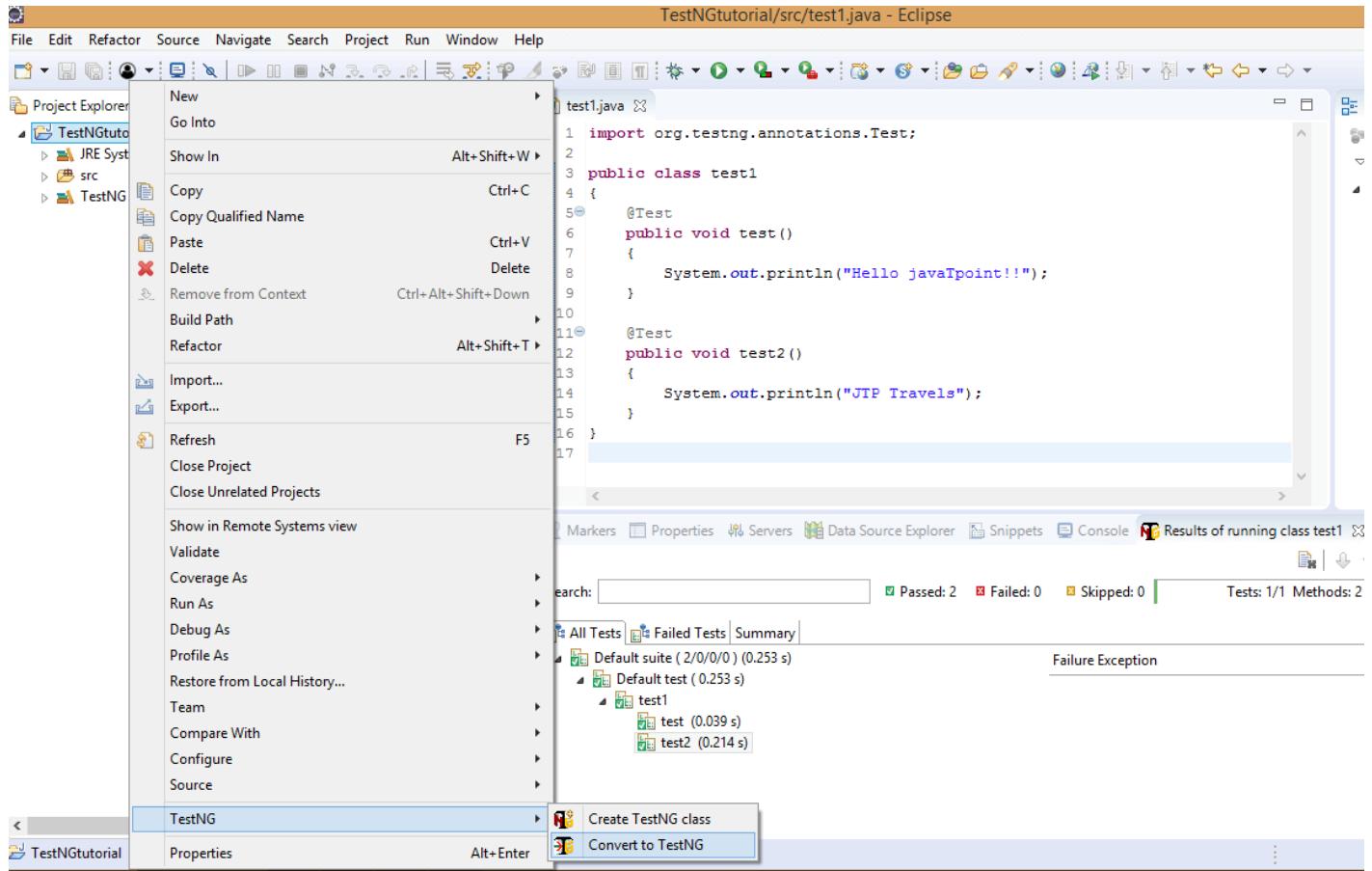
The screenshot shows the Eclipse IDE interface. The title bar reads "TestNGtutorial/src/test1.java - Eclipse". The menu bar includes File, Edit, Refactor, Source, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Cut, Copy, Paste, Find, and Run. The Project Explorer view on the left shows a project named "TestNGtutorial" with a "src" folder containing a "default package" and a file named "test1.java". The code editor on the right displays the following Java code:

```
1 import org.testng.annotations.Test;
2
3 public class test1
4 {
5 @Test
6 public void test()
7 {
8 System.out.println("Hello javaTpoint!!!");
9 }
10 @Test
11 public void test2()
12 {
13 System.out.println("JTP Travels");
14 }
15 }
16
17
```

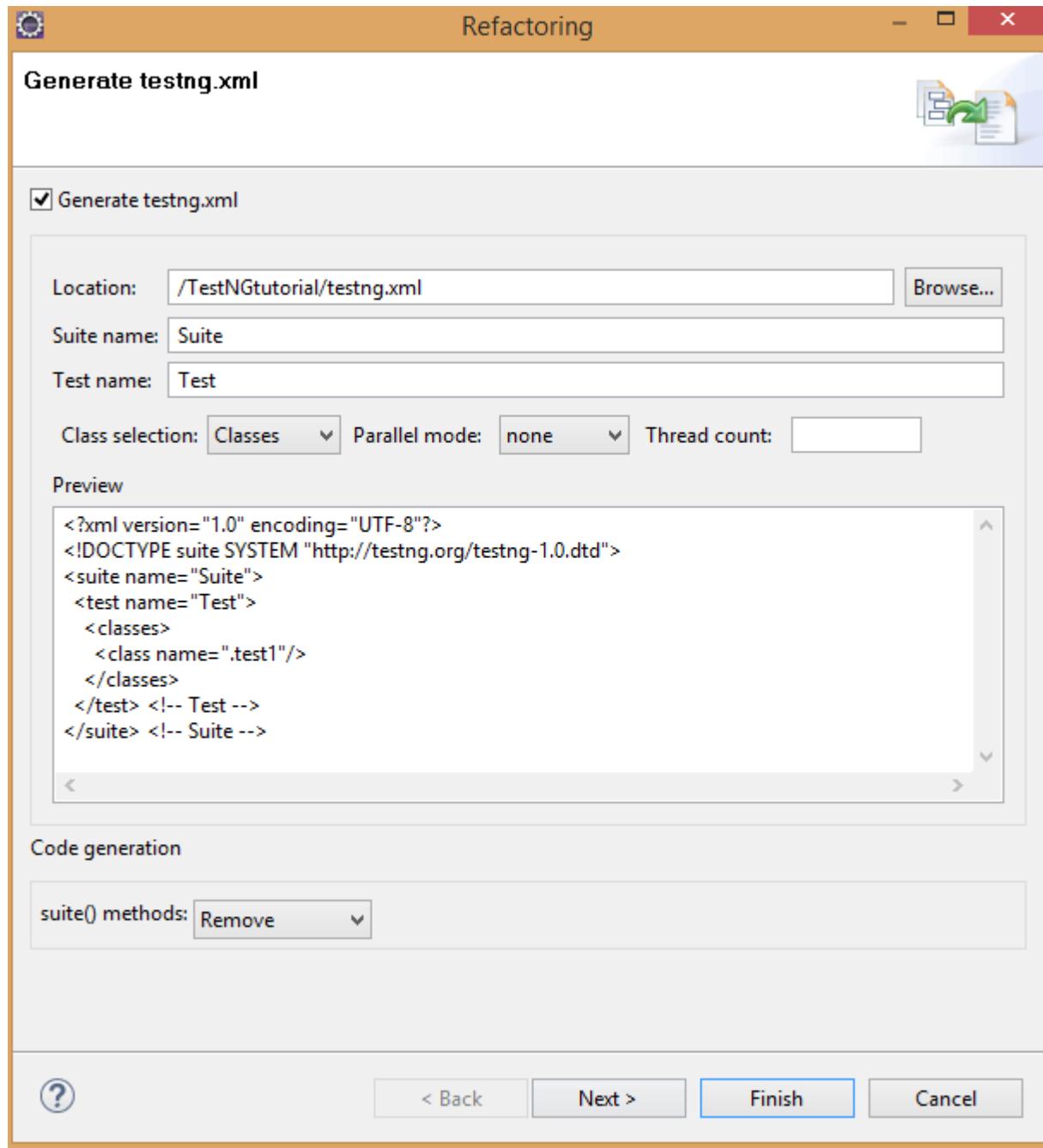
## Output

## How to create a xml file

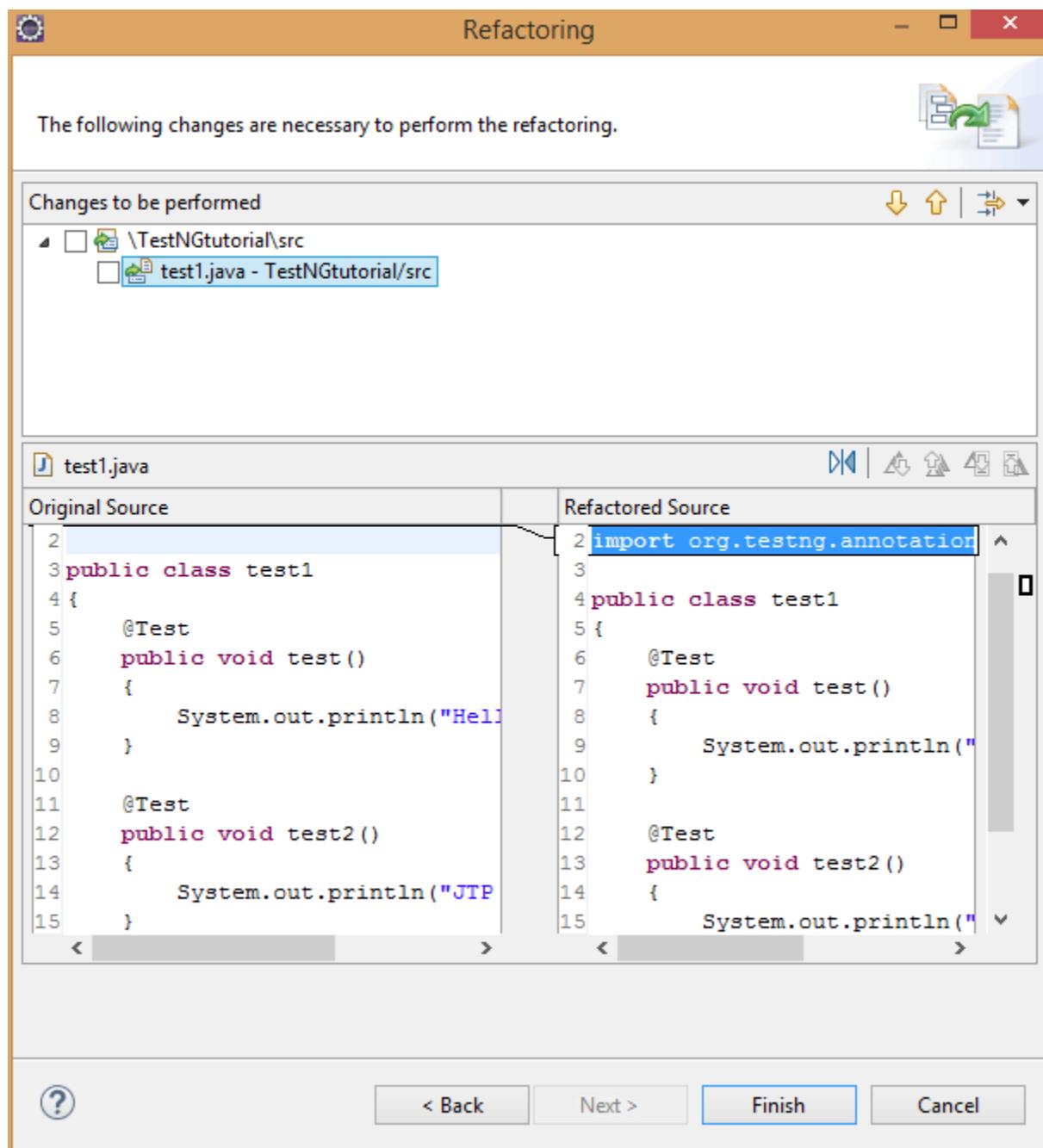
- Right click on the project. Move your cursor down, and you will see TestNG and then click on the **Convert to TestNG**.



- The below screen shows the preview of the xml file. Click on the **Next** button.



- Click on the **Finish** button.



- o The testing.xml file is shown below:



The screenshot shows a code editor window with two tabs: "test1.java" and "testing.xml". The "testing.xml" tab is active, displaying the following XML code:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3<suite name="Suite">
4<test name="Test">
5<classes>
6 <class name=".test1"/>
7 </classes>
8 </test> <!-- Test -->
9 </suite> <!-- Suite -->
10 |
```

The code defines a TestNG suite named "Suite" containing a single test named "Test". The test class is ".test1". The "Source" tab is selected at the bottom of the editor.

In the above source code of xml file, suite is at the higher hierarchy in TestNG. Inside the , you have to define the test name folder. This test name folder is the name of the folder. For example, In a loan company, there are three different types of modules such as personal loan, home loan and car loan, and each module contain its own test cases. All these test cases are defined in the test name folder.

Now we will create the module of personal loan.

**Step 1:** We first create two java files and both the files contain test cases.

### tes1.java

```
1. package day1;
2. import org.testng.annotations.Test;
3.
4. public class module1
5. {
6. @Test
7. public void test1()
8. {
9. System.out.println("Hello javaTpoint!!");
10. }
11.
12. @Test
```

```
13. public void test2()
14. {
15. System.out.println("JTP Travels");
16. }
```

**test2.java**

```
1. package day1;
2.
3. import org.testng.annotations.Test;
4.
5. public class module2
6. {
7. @Test
8. public void test3()
9. {
10. System.out.println("hindi100.com");
11. }
12. }
```

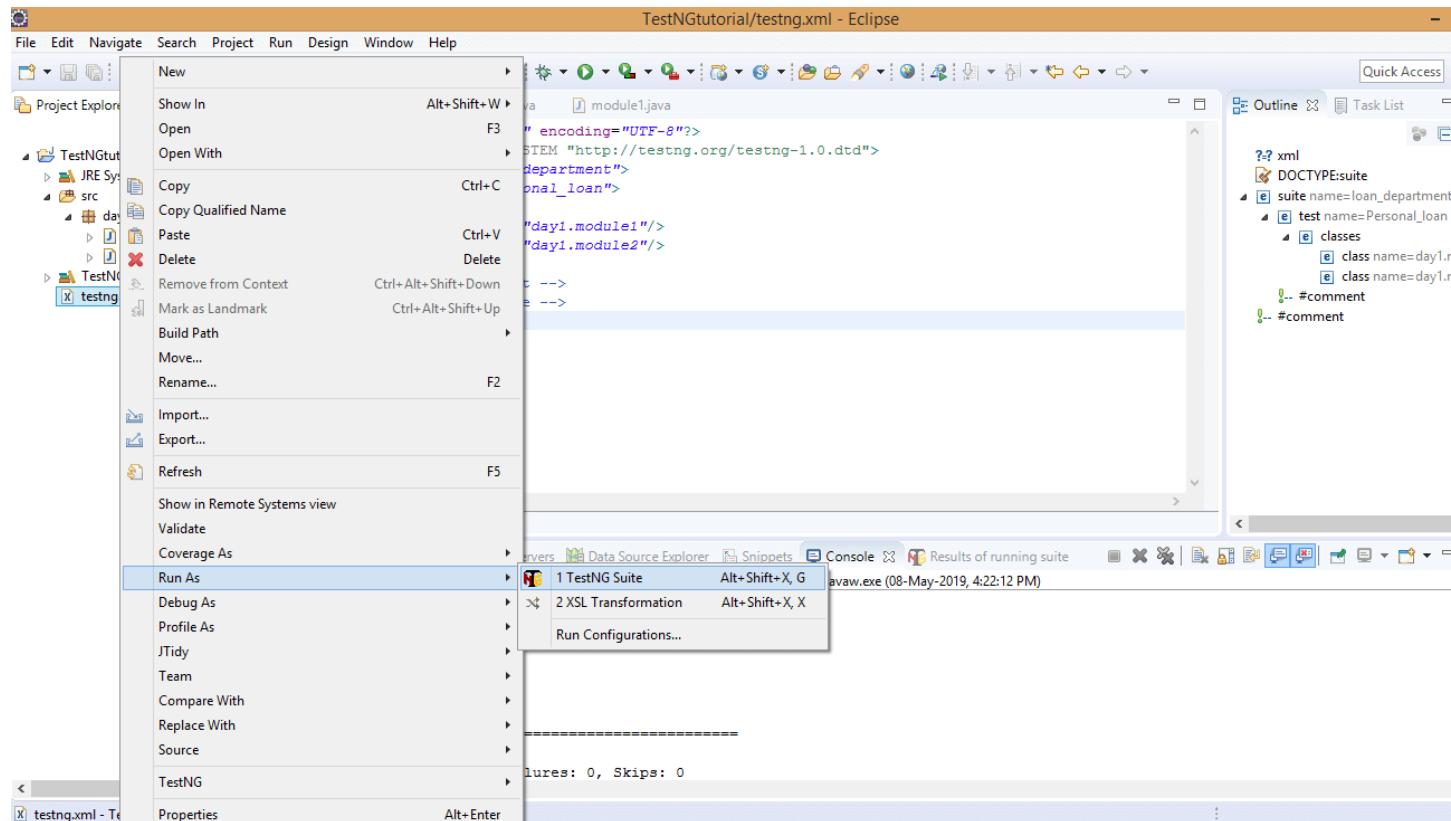
**Step 2:** Now we will create the xml file.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3. <suite name="loan_department">
4. <test name="Personal_loan">
5. <classes>
6. <class name="day1.module1"/>
7. <class name="day1.module2"/>
8. </classes>
9. </test> <!-- Test -->
10. </suite> <!-- Suite -->
```

In the above XML file, we have created the suite "**loan\_department**". We have created the module "Personal loan" inside the suite and within this module, we have created the test cases defined in the classes day1.module1 and day1.module2, where day1 is the package name and module1 and module2, are the classes.

**Step 3:** In this step, we will run the test cases. Now we do not need to run the java files individually. We have to run the XML file which will automatically execute all the test cases as we have configured all the class files inside the XML file that are containing test cases.

Right click on the **testng.xml** file and then move down to the **Run As** and then click on the **1 TestNG Suite**.



## Output

```
Design Source
Markers Properties Servers Data Source Explorer Snippets Console Results of running suite
<terminated> TestNGtutorial_testng.xml [TestNG] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (08-May-2019, 4:38:46 PM)
[TestNG] Running:
D:\TestNGtutorial\testng.xml

Hello javaTpoint!!
JTP Travels
hindi100.com

=====
loan_department
Total tests run: 3, Failures: 0, Skips: 0
=====
```



**UNIT I FOUNDATIONS OF SOFTWARE TESTING**

Why do we test Software?, Black-Box Testing and White-Box Testing, Software Testing Life Cycle, V-model of Software Testing, Program Correctness and Verification, Reliability versus Safety, Failures, Errors and Faults (Defects), Software Testing Principles, Program Inspections, Stages of Testing: Unit Testing, Integration Testing, System Testing

**UNIT-I / PART-A**

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>Mention the objectives of software testing. (Nov 16, 17, Apr 22)</b><br>Objectives of Software testing are: <ul style="list-style-type: none"> <li>❖ Finding defects which may get created by the programmer while developing the software.</li> <li>❖ Gaining confidence in and providing information about the level of quality.</li> <li>❖ To prevent defects.</li> <li>❖ To make sure that the end result meets the business and user requirements.</li> <li>❖ To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.</li> <li>❖ To gain the confidence of the customers by providing them a quality product.</li> </ul> |
| 2. | <b>Define process.</b><br>Process in the software engineering domain, is the set of methods, practices, standards, documents, activities, policies, and procedures that software engineers use to develop and maintain a software system and its associated artifacts, such as project ,test plans, design documents, code, and manuals.                                                                                                                                                                                                                                                                                                                                                                   |
| 3. | <b>Differentiate between verification and validation.</b><br>Validation is the process of evaluating a software system or component during, or at the end of, the development cycle in order to determine whether it satisfies specified requirements. Validation is usually associated with traditional execution-based testing, that is, exercising the code with test cases.<br>Verification is the process of evaluating a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.                                                                                                                      |
| 4. | <b>What is testing and debugging?</b><br>Testing is generally described as a group of procedures carried out to evaluate some aspect of a piece of software. Testing can be described as a process used for revealing defects in software, and for establishing that the software has attained a specified degree of quality with respect to selected attributes. Debugging, or fault localization is the process of (1) locating the fault or defect, (2) repairing the code, and (3) retesting the code                                                                                                                                                                                                  |
| 5. | <b>List the benefits of test process improvement.</b> <ul style="list-style-type: none"> <li>❖ Smarter testers</li> <li>❖ Higher quality software</li> <li>❖ The ability to meet budget and scheduling goals</li> <li>❖ Improved planning</li> <li>❖ Ability to meet quantifiable testing goals</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6.  | <p><b>Define Maturity goals.</b></p> <p>The maturity goals identify testing improvement goals that must be addressed in order to achieve maturity at that level. To be placed at a level, an organization must satisfy the maturity goals at that level.</p>                                                                                                                                                                                                                                                                                                                                                   |
| 7.  | <p><b>Define Activities, tasks and responsibilities (ATR).</b></p> <p>The ATRs addresses implementation and organizational adaptation issues at each TMM level. Supporting activities and tasks are identified, and responsibilities are assigned to appropriate groups. Responsibilities are the works or the tasks that are assigned to the group or individuals. Tasks are the goals or the requirements that needsto be completed in a stipulated time. Activities are the work plan or the blueprint that needs to be followed to reach the completion of the task that the group is responsible for.</p> |
| 8.  | <p><b>List the three critical views (CV).</b></p> <p>Definition of tester's roles is essential in developing a maturity framework.</p> <ul style="list-style-type: none"> <li>❖ The manager's view involves commitment and ability to perform activities and tasks related toimproving testing capability.</li> <li>❖ The developer/tester's view encompasses the technical activities and tasks that, when applied,constitute quality testing practices.</li> <li>❖ The user's or client's view is defined as a cooperating, or supporting, view.</li> </ul>                                                  |
| 9.  | <p><b>List out the levels of the TMM. (Apr 18,22)</b></p> <p>Level 1: Initial</p> <p>Level 2: Phase Define</p> <p>Level 3: Integration</p> <p>Level 4: Management and Measurement</p> <p>Level 5: Optimization/Defect Prevention and Quality Control</p>                                                                                                                                                                                                                                                                                                                                                       |
| 10. | <p><b>Compare Errors Vs Faults (Defects) Vs Failures.</b></p> <p>An error is a mistake, misconception, or misunderstanding on the part of a software developer. A fault (defect) is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification. Faults or defects are sometimes called -bugs.   A failure is the inability of a software system or component to perform its required functions within specified performance requirements.</p>                                                    |
| 11. | <p><b>How will the fault manifest itself as a failure?</b></p> <ul style="list-style-type: none"> <li>❖ The input to the software must cause the faulty statement to be executed.</li> <li>❖ The faulty statement must produce a different result than the correct statement. This event produces anincorrect internal state for the software.</li> <li>❖ The incorrect internal state must propagate to the output, so that the result of the fault is observable.</li> </ul>                                                                                                                                 |
| 12. | <p><b>Tell about test, test Oracle and Test Bed. (Nov 17,Apr 18)</b></p> <p>A test is a group of related test cases, or a group of related test cases and test procedures (steps needed to carry out a test. A test oracle is a document, or piece of software that allows testers to determine whether a test has been passedor failed. A test bed is an environment that contains all the hardware and software needed to test a software component or a software system.</p>                                                                                                                                |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13. | <p><b>What are the contents of a test case?</b></p> <ul style="list-style-type: none"> <li>❖ A test case in a practical sense is a test-related item which contains the following information:</li> <li>❖ A set of test inputs: These are data items received from an external source by the code under test. The external source can be hardware, software, or human.</li> <li>❖ Execution conditions: These are conditions required for running the test, for example, a certain state of a database, or a configuration of a hardware device.</li> <li>❖ Expected outputs: These are the specified results to be produced by the code under test.</li> </ul> |
| 14. | <p><b>Define Software Quality.</b></p> <p>Two concise definitions for quality are found in the IEEE Standard Glossary of Software Engineering Terminology:</p> <ul style="list-style-type: none"> <li>❖ Quality relates to the degree to which a system, system component, or process meets specified requirements.</li> <li>❖ Quality relates to the degree to which a system, system component, or process meets customer or user needs, or expectations.</li> </ul>                                                                                                                                                                                          |
| 15. | <p><b>Define metric and quality metric.</b></p> <p>A metric is a quantitative measure of the degree to which a system, system component, or process possesses a given attribute. A quality metric is a quantitative measurement of the degree to which an item possesses a given quality attribute.</p>                                                                                                                                                                                                                                                                                                                                                         |
| 16. | <p><b>List the Quality Attributes.</b></p> <ul style="list-style-type: none"> <li>❖ Correctness</li> <li>❖ Reliability</li> <li>❖ Usability</li> <li>❖ Integrity</li> <li>❖ Portability</li> <li>❖ Maintainability</li> <li>❖ Interoperability.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                      |
| 17. | <p><b>Define Software Quality Assurance Group.</b></p> <p>The software quality assurance (SQA) group is a team of people with the necessary training and skills to ensure that all necessary actions are taken during the development process so that the resulting software conforms to established technical requirements.</p>                                                                                                                                                                                                                                                                                                                                |
| 18. | <p><b>What are the sources of defects? (Apr 17)</b></p> <p>Defects have detrimental effects on software users, and software engineers work very hard to produce high-quality software with a low number of defects. Defects occur from the following sources: Lack of Education, Poor communication, Oversight, Transcription, Immature process.</p>                                                                                                                                                                                                                                                                                                            |
| 19. | <p><b>Can you Classify defect prevention strategies?</b></p> <p>Software Defects/ Bugs are normally classified as per:</p> <ul style="list-style-type: none"> <li>❖ Severity / Impact</li> <li>❖ Probability / Visibility</li> <li>❖ Priority / Urgency</li> </ul> <p>Related Dimension of Quality Related Module / Component</p> <ul style="list-style-type: none"> <li>❖ Phase Detected and Phase Injected.</li> </ul>                                                                                                                                                                                                                                        |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20. | <p><b>Define fault with an example. (Nov 16,Apr 19)</b></p> <ul style="list-style-type: none"> <li>❖ A fault (defect) is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification. Faults or defects are sometimes called bugs. For example, a software engineer who did not understand the precedence order of operators could inject a defect in an equation.</li> <li>❖ Defects are variance between expected and actual. Defect is an error found after the application goes into production. It commonly refers to several troubles with the software products, with its external behavior or with its internal features. Example: The inability of a software system or component to perform its required functions within specified performance requirements (Failure), A requirement of the customer that was not fulfilled.</li> </ul> |
| 21. | <p><b>What is design defect?</b></p> <p>A design defect is a problem with the product's design that makes the product inherently dangerous or useless, even if it is manufactured perfectly and made of the best-quality materials.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 22. | <p><b>What are the Design Defect Classes?(Nov 21)</b></p> <p>Design defects occur when system components, interactions between system components, interactions between the components and outside software/hardware, or users are incorrectly designed. This covers Algorithmic and processing, Control, logic and sequence, Data, Module interface description,External interface description</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 23. | <p><b>What are the Coding Defect Classes?</b></p> <p>Coding defects are derived from errors in implementing the code. Coding defects classes are closely related to design defect classes especially if pseudo code has been used for detailed design.</p> <ul style="list-style-type: none"> <li>❖ Algorithmic and processing</li> <li>❖ Control, logic, and sequence</li> <li>❖ Typographical data flow</li> <li>❖ Data</li> <li>❖ Module interface</li> <li>❖ Code documentation</li> <li>❖ External hardware,</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 24. | <p><b>What is meant by feature defect? (Nov 18)</b></p> <p>Features may be described as distinguishing characteristics of a software component or system. Feature defects are due to feature descriptions that are missing, incorrect, incomplete, or superfluous.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 25. | <p><b>What is Defect Repository?</b></p> <p>Defects can be classified in many ways. It is important for an organization to follow a single classification scheme and apply it to all projects. The defect types and frequency of occurrence should be used in test planning, and test design. Execution-based testing strategies should be selected that have the strongest possibility of detecting particular types of defects. Defect Repositories contain the information of the various types of defects that can happen or have occurred previously. These also contain information on the protocol that needs to be followed to eradicate or prevent defects.</p>                                                                                                                                                                                                                                                                                        |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26. | <b>What is a test case? (Nov 21)</b><br>A test case is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.                                                                                                                                                                                              |
| 27. | <b>What is the V&amp;V task in testing phase?</b><br>V and V stands for verification and validation. It includes execution of system test case, Execution of acceptance test case, Updating of traceability metrics and Risk analysis.                                                                                                                                                                                                                    |
| 28. | <b>Mention the role of test engineer in software development organization. (Apr 17, Nov 21)</b><br>The tester's job is to reveal defects, find weak points, inconsistent behavior, and circumstances where the software does not work as expected. Goal of a tester is to work with the developers to produce high-quality software that meets the customers requirements.                                                                                |
| 29. | <b>Why test cases should be developed for both valid and invalid inputs? (Apr 19)</b><br>Use of test cases that are based on invalid inputs is very useful for revealing defects since they may exercise the code in unexpected ways and identify unexpected software behavior. Invalid inputs also help developers and Software Test Engineers to evaluate the robustness of the software, that is, its ability to recover when unexpected events occur. |
| 30. | <b>Mention the role of process in software quality. (Nov 18)</b><br>The practices that contribute to the development of high-quality software are project planning, requirements management, development of formal specifications, structured design with use of information hiding and encapsulation, design and code reuse, inspections and reviews, product and process measures, education and training of software professionals.                    |
| 31. | <b>Mention the relationship between testing effectiveness and the quality of software system. (Apr 21)</b><br>The relationship between testing effectiveness and quality of software is about the activities designed to make sure the project is conforming to the expectations of the stakeholders, while test is a process to explore a system to find defects.                                                                                        |
| 32. | <b>Who is called as a Test Specialist? (Apr 21)</b><br>A test specialist is responsible for evaluating and running diagnostic tests for system networks and applications to ensure stability and efficiency according to the quality standards. Test specialists analyse the system's features and write findings reports for improvement and technical resolutions                                                                                       |

**UNIT-I / PART-B**

|    |                                                                                                                                                                                                                                  |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | Analyze the Role of process in Software quality.                                                                                                                                                                                 |
| 2. | Summarize the difficulties and challenges faced by a tester? Explain in detail. (Apr 22)                                                                                                                                         |
| 3. | Explain 5 level structure of testing maturity model in detail. (Apr 18)                                                                                                                                                          |
| 4. | Give an overview of test related activities that should be done for V-model architecture. (or) Describe the test related activities using V model in requirement specification, design, coding and installation phases. (Apr 18) |
| 5. | Explain software testing principles in detail. (Nov 16, 17, 18, 21, Apr 17, 18, 19, 21)                                                                                                                                          |
| 6. | Analyze tester's role in software development organization. (Nov 16, 17, Apr 18)                                                                                                                                                 |
| 7. | Explain in detail processing and monitoring of the defects with defect repository. (Nov 16, Apr 18)                                                                                                                              |

|     |                                                                                                                                                                                              |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8.  | Discuss about all the defects related to requirement specification, design, coding and testing phases.                                                                                       |
| 9.  | Explain various design defects with suitable examples. ( <i>Nov 17,Apr 21</i> )                                                                                                              |
| 10. | What are the typical origins of defects? Explain the major classes of defects in the software artifacts. (or) Write short notes on origin of defects. ( <i>Nov 18,Apr 17,22</i> )            |
| 11. | Describe about tester support for developing a defect repository. ( <i>Nov 18</i> )                                                                                                          |
| 12. | Why it is important to meticulously inspect test result and discover the drawbacks incase if you fail to inspect? Illustrate with example? ( <i>Nov 21</i> )                                 |
| 13. | i) Discuss in detail about various types of testing axioms.<br>ii) Discuss in detail Tester's role in software development organization ( <i>Apr 21</i> )                                    |
| 14. | Suppose you are testing defect coin problem artifacts, Identify the causes of various defects. What steps could have been taken to prevent the various classes of defects? ( <i>Apr 22</i> ) |

**UNIT II TEST PLANNING 6**

The Goal of Test Planning, High Level Expectations, Intergroup Responsibilities, Test Phases, Test Strategy, Resource Requirements, Tester Assignments, Test Schedule, Test Cases, Bug Reporting, Metrics and Statistics.

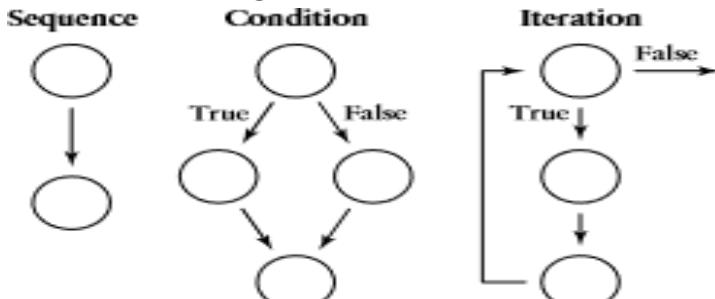
**UNIT-II / PART-A**

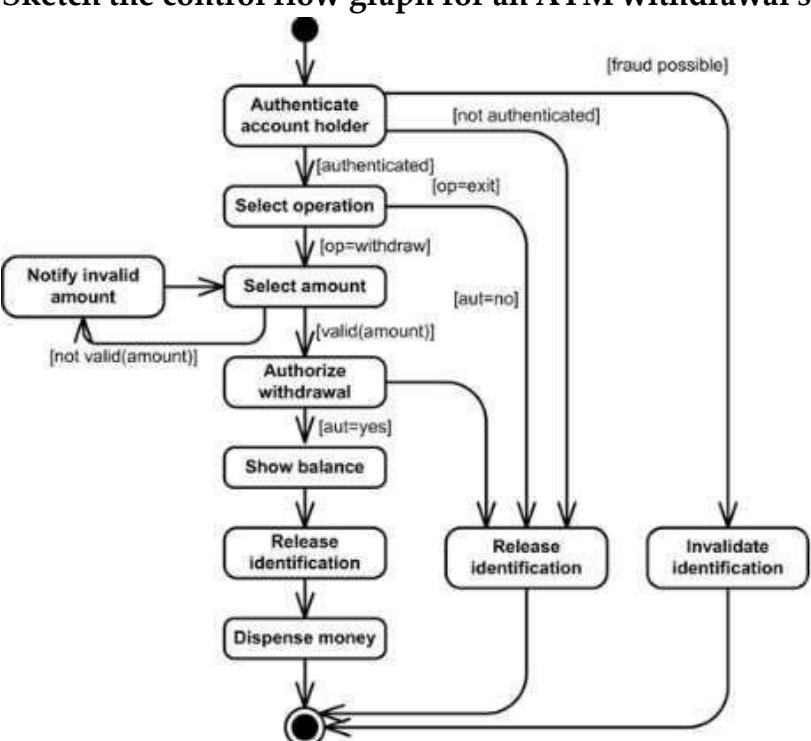
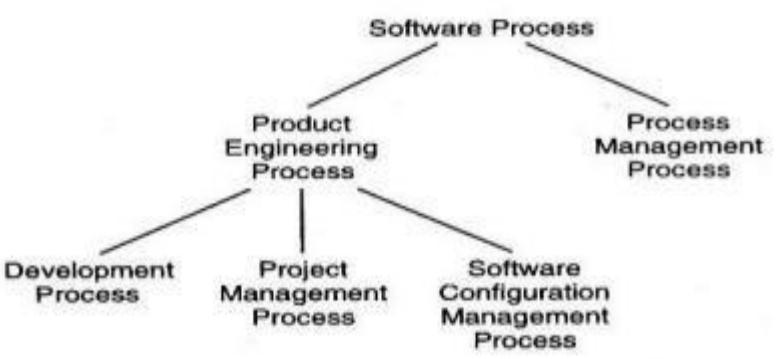
|    |                                                                                                                                                                                                                                                                                                                                                                                                  |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>List Black box knowledge source and testing methods.</b><br>Black Box knowledge is the information or knowledge about the Sources, Requirements, document, Specifications, Domain knowledge, Defect analysis and Data.<br>Some of the Black Box methods are:<br>Equivalence class, partitioning, Boundary value analysis, State transition testing, Cause and effect graphing, Error guessing |
| 2. | <b>List white box knowledge source and testing methods.</b><br>White Box knowledge is the information about High-level design, detailed design, Control flow graphs, Cyclomatic Complexity. Some of the White Box methods are:<br>Statement testing, Branch testing, Path testing, Data flow testing, Mutation testing, Loop testing                                                             |
| 3. | <b>Compare black box and white box testing. (<i>Nov 17,18,Apr 19</i>)</b>                                                                                                                                                                                                                                                                                                                        |
|    | <b>Black box testing</b>                                                                                                                                                                                                                                                                                                                                                                         |
|    | Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program.                                                                                                                                                                                                                                                   |
|    | This type of testing is carried out by Testers                                                                                                                                                                                                                                                                                                                                                   |
|    | Implementation Knowledge is not required to carry out Black Box Testing                                                                                                                                                                                                                                                                                                                          |
|    | <b>White box testing</b>                                                                                                                                                                                                                                                                                                                                                                         |
|    | White box testing is the software testing method in which internal structure is being known to tester who is going to test the software                                                                                                                                                                                                                                                          |
|    | Testing is carried out by software developers                                                                                                                                                                                                                                                                                                                                                    |
|    | Implementation Knowledge is required to carry out White Box Testing.                                                                                                                                                                                                                                                                                                                             |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4. | <p><b>Define a State and what is meant by finite-state machine?</b></p> <p>A state is an internal configuration of a system or component. It is defined in terms of the values assumed at a particular time for the variables that characterize the system or component.</p> <p>Finite-state machine is an abstract machine that can be represented by a state graph having a finite number of states and a finite number of transitions between states.</p>                                                                                                                                                                                                                                                                                                                                                                        |
| 5. | <p><b>Define Random Testing.(Apr 22)</b></p> <p>Each software module or system has an input domain from which test input data is selected. If a tester randomly selects inputs from the domain, this is called random testing.</p> <p><b>Example:</b></p> <p>If the valid input domain for a module is all positive integers between 1 and 100, the tester using this approach would randomly, or unsystematically, select values from within that domain; for example, the values 55, 24, 3 might be chosen</p>                                                                                                                                                                                                                                                                                                                    |
| 6. | <p><b>Show the need of State Transition testing in test case design.</b></p> <p>State transition testing is useful for both procedural and object-oriented development. It is based on the concepts of states and finite-state machines, and allows the tester to view the developing software in term of its states, transitions between states, and the inputs and events that trigger state changes.</p>                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 7. | <p><b>List the advantages of Equivalence class partitioning.</b></p> <p>If a tester is viewing the software-under-test as a black box with well-defined inputs and outputs, a good approach to selecting test inputs is to use a method called equivalence class partitioning. Equivalence class partitioning results in a partitioning of the input domain of the software under test.</p> <p>Advantages of Equivalence Class Partitioning are,</p> <ul style="list-style-type: none"> <li>❖ It eliminates the need for exhaustive testing, which is not feasible.</li> <li>❖ It guides a tester in selecting a subset of test inputs with a high probability of detecting a defect.</li> <li>❖ It allows a tester to cover a larger domain of inputs/outputs with a smaller subset selected from an equivalence class.</li> </ul> |
| 8. | <p><b>What are the steps involved in Equivalence Class Partitioning?</b></p> <p>A good approach includes the following steps.</p> <ul style="list-style-type: none"> <li>❖ Each equivalence class should be assigned a unique identifier. A simple integer is sufficient.</li> <li>❖ Develop test cases for all valid equivalence classes until all have been covered by (included in) a test case. A given test case may cover more than one equivalence class.</li> <li>❖ Develop test cases for all invalid equivalence classes until all have been covered individually. This is to insure that one invalid case does not mask the effect of another or prevent the execution of another.</li> </ul>                                                                                                                            |
| 9. | <p><b>What is test set and test data set?</b></p> <p>A test set T is said to be mutation adequate for program P provided that for every inequivalent mutant <math>P_i</math> of P there is an element <math>t</math> in T such that <math>P_i(t)</math> is not equal to <math>P(t)</math>. A test data set is statement, or branch, adequate if a test set T for program P causes all the statements, or branches, to be executed respectively.</p>                                                                                                                                                                                                                                                                                                                                                                                 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10. | <p><b>Discuss about rules-of-thumb.</b></p> <ul style="list-style-type: none"> <li>❖ The rules-of-thumb described below are useful for getting started with boundary value analysis.</li> <li>❖ If an input condition for the software-under-test is specified as a range of values, develop valid test cases for the ends of the range, and invalid test cases for possibilities just above and below the ends of the range.</li> <li>❖ If an input condition for the software-under-test is specified as a number of values, develop valid test cases for the minimum and maximum numbers as well as invalid test cases that include one lesser and one greater than the maximum and minimum.</li> <li>❖ If the input or output of the software-under-test is an ordered set, such as a table or a linear list, develop tests that focus on the first and last elements of the set.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 11. | <p><b>List the steps in developing test cases with a cause-and-effect graph.</b></p> <ul style="list-style-type: none"> <li>❖ The tester must decompose the specification of a complex software component into lower-level units.</li> <li>❖ For each specification unit, the tester needs to identify causes and their effects. A cause is a distinct input condition or an equivalence class of input conditions. An effect is an output condition or a system transformation. Putting together a table of causes and effects helps the tester to record the necessary details. The logical relationships between the causes and effects should be determined. It is useful to express these in the form of a set of rules.</li> <li>❖ From the cause-and-effect information, a Boolean cause-and-effect graph is created. Nodes in the graph are causes and effects. Causes are placed on the left side of the graph and effects on the right. Logical relationships are expressed using standard logical operators such as AND, OR, and NOT, and are associated with arcs.</li> <li>❖ The graph may be annotated with constraints that describe combinations of causes and/or effects that are not possible due to environmental or syntactic constraints.</li> <li>❖ The graph is then converted to a decision table.</li> <li>❖ The columns in the decision table are transformed into test cases.</li> </ul> |
| 12. | <p><b>List the application scope of adequacy criteria.</b></p> <ul style="list-style-type: none"> <li>❖ Helping testers to select properties of a program to focus on during test;</li> <li>❖ Helping testers to select a test data set for a program based on the selected properties;</li> <li>❖ Supporting testers with the development of quantitative objectives for testing;</li> <li>❖ Indicating to testers whether or not testing can be stopped for that program.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 13. | <p><b>What are Loop testing strategies?</b></p> <p>Loop testing strategies focus on detecting common defects associated with these structures. For example, in a simple loop that can have a range of zero to n iterations, test cases should be developed so that there are:</p> <ol style="list-style-type: none"> <li>1. zero iterations of the loop, i.e., the loop is skipped in its entirety;</li> <li>2. one iteration of the loop;</li> <li>3. two iterations of the loop;</li> <li>4. k iterations of the loop where k &lt; n;</li> <li>5. n - 1 iterations of the loop;</li> <li>6. n iterations of the loop (if possible).</li> </ol>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14. | <p><b>What is Cyclomatic complexity? Give a note on the procedure to compute cyclomatic complexity. (Nov 16) (or) Write the formula for cyclomatic complexity. (Apr 18)</b></p> <ul style="list-style-type: none"> <li>❖ The cyclomatic complexity of a section of source code is the number of linearly independent paths within it</li> <li>❖ Mathematically, the cyclomatic complexity of a structured program is defined with reference to the control flow graph of the program, a directed graph containing the basic blocks of the program, with an edge between two basic blocks if control may pass from the first to the second.</li> <li>❖ The complexity M is then defined as</li> </ul> $M = E - N + 2P,$ <p>Where</p> <p>E = the number of edges of the graph.</p> <p>N = the number of nodes of the graph.</p> <p>P = the number of connected components.</p>    |
| 15. | <p><b>What is Mutation testing?</b></p> <p>Mutation testing is another approach to test data generation that requires knowledge of code structure, but it is classified as a fault-based testing approach. It considers the possible faults that could occur in a software component as the basis for test data generation and evaluation of testing effectiveness.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 16. | <p><b>List the two major assumptions of Mutation testing.</b></p> <p>Mutation testing makes two major assumptions:</p> <ol style="list-style-type: none"> <li>1. The competent programmer hypothesis. This states that a competent programmer writes programs that are nearly correct. Therefore we can assume that there are no major construction errors in the program; the code is correct except for a simple error(s).</li> <li>2. The coupling effect. This effect relates to questions a tester might have about how well mutation testing can detect complex errors since the changes made to the code are very simple. DeMillo states that test data that can distinguish all programs differing from a correct one only by simple errors are sensitive enough to distinguish it from programs with more complex errors.</li> </ol>                                   |
| 17. | <p><b>Point out the advantages and disadvantages of random testing.</b></p> <p><b>Advantages of Random Testing:</b></p> <ul style="list-style-type: none"> <li>❖ Random testing gives us an advantage of easily estimating software reliability from test outcomes.</li> <li>❖ They're done from a user's point of view</li> <li>❖ Do not require to know programming languages or how the software has been implemented</li> <li>❖ Can be conducted by a body independent from the developers,</li> </ul> <p><b>Disadvantages of random testing:</b></p> <ul style="list-style-type: none"> <li>❖ They are not realistic &amp; many of the tests are redundant and unrealistic.</li> <li>❖ More time is spent on analyzing results.</li> <li>❖ One cannot recreate the test if data is not recorded which was used for testing.</li> <li>❖ False positive can occur</li> </ul> |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18. | <p><b>What is a control flow graph?(Nov 21)</b></p> <p>The control flow graph <math>G = (N, E)</math> of a program consists of a set of nodes <math>N</math> and a set of edges <math>E</math>. Each node represents a set of program statements. There are five types of nodes. There is a unique entry node and a unique exit node. There is an edge from node <math>n_1</math> to node <math>n_2</math> if the control may flow from the last statement in <math>n_1</math> to the first statement in <math>n_2</math>.</p>                                                                                                                                                                                                                                                                                     |
| 19. | <p><b>Discuss about Desk checking.</b></p> <p>Desk checking is a paper and pencil exercise where the programmer works through the program by hand keeping track of the values of each variable and the statements that are executed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 20. | <p><b>What do you mean by test coverage?</b></p> <p>Test coverage measures the amount of testing performed by a set of tests. Whenever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and it is known as test coverage. The basic coverage measure is where the coverage item is whatever we have been able to count and see whether a test has exercised or used this item.</p> <p>Coverage = (Number of Coverage items exercised / Total number of coverage items) * 100</p>                                                                                                                                                                                                                                                    |
| 21. | <p><b>Mention the ways by which test cases may be generated. Generate a test case for a scenario.(Apr 17,22)</b></p> <ul style="list-style-type: none"> <li>❖ Goal-oriented approach, Random approach, Specification-based technique, Source-code-based technique, Sketch-diagram-based approach</li> <li>❖ Scenario is thread of operations whereas Test cases are set of input and output given to the System.</li> </ul> <p>For example: Checking the functionality of Login button is Test scenario and Test Cases for this Test Scenario are:</p> <ol style="list-style-type: none"> <li>1. Click the button without entering user name and password.</li> <li>2. Click the button only entering User name.</li> <li>3. Click the button while entering wrong user name and wrong password, etc...</li> </ol> |
| 22. | <p><b>Define COTS components. (Nov 18)</b></p> <p>The reusable component may come from a code reuse library within their org or, as is most likely, from an outside vendor who specializes in the development of specific types of software components. Components produced by vendor org are known as commercial off-the shelf, or COTS, components.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 23. | <p><b>What are the basic primes that are used in a structured program? (Nov 17)</b></p> <p>There are three basic primitives—sequential (e.g., assignment statements), decision (e.g., if/then/else statements), and iterative (e.g., while, for loops). Graphical representations for these three primitives are shown in Figure.</p>                                                                                                                                                                                                                                                                                                                                                                                          |

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24. <b>What are the factors affecting less than 100% degree of coverage? (Apr 18)</b><br>The planned degree of coverage may be less than 100% possibly due to the following: <ul style="list-style-type: none"><li>❖ The nature of the unit, some statements/branches may not be reachable. The unit may be simple, and not mission, or safety, critical, and so complete coverage is thought to be unnecessary.</li><li>❖ The lack of resources, the time set aside for testing is not adequate to achieve 100% coverage. There are not enough trained testers to achieve complete coverage for all of the units. There is a lack of tools to support complete coverage.</li><li>❖ Other project-related issues such as timing, scheduling, and marketing constraints</li></ul>                                                                                                                                                                               |
| <b>25. Sketch the control flow graph for an ATM withdrawal system. (Nov 16)</b><br> <pre> graph TD     Start(( )) --&gt; Auth[Authenticate account holder]     Auth -- "[not authenticated]" --&gt; SelectOp[Select operation]     SelectOp -- "[op=exit]" --&gt; End(( ))     SelectOp -- "[op=withdraw]" --&gt; SelectA[Select amount]     SelectA -- "[valid(amount)]" --&gt; Authorize[Authorize withdrawal]     Authorize -- "[aut=yes]" --&gt; ShowBal[Show balance]     ShowBal --&gt; ReleaseI1[Release identification]     ReleaseI1 --&gt; Dispense[Dispense money]     Dispense --&gt; End     SelectA -- "[not valid(amount)]" --&gt; Notify[Notify invalid amount]     Notify --&gt; SelectA     Authorize -- "[aut=no]" --&gt; ReleaseI2[Release identification]     ReleaseI2 --&gt; Invalidate[Invalidate identification]     Invalidate --&gt; End   </pre> |
| <b>26. What do you meant by test adequacy criteria? (Apr 19)</b><br>A test adequacy criterion is a predicate that is true (satisfied) or false (not satisfied) of a(program, test suite)pair. Usually a test adequacy criterion is expressed in the form of a rule for deriving a set of test obligations from another artifact, such as a program or specification. The adequacy criterion is then satisfied if every test obligation is satisfied by at least one test case in the suite                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>27. Give the diagrammatic representation of various components of engineered processes.(Apr 21)</b><br> <pre> graph TD     PMP[Project Management Process] --&gt; DP[Development Process]     PMP --&gt; SCMP[Software Configuration Management Process]     PMP --&gt; SP[Software Process]     SP --&gt; PE[Product Engineering Process]     SP --&gt; PMP2[Process Management Process]   </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

|     |                                                                                                                                                                                                                                                                                                                                 |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28. | <b>What is code complexity testing?(Apr 21)</b><br>It is a quantitative measure of the number of linearly independent paths through a program's source code. It was developed by Thomas J. McCabe, Sr. in 1976. Cyclomatic complexity may also be applied to individual functions, modules, methods or classes within a program |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**UNIT-II / PART-B**

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  | Show how black box testing is performed in COTS components?                                                                                                                                                                                                                                                                                                                                                                                                |
| 2.  | With suitable example describe how cause-and -effect graphing, and state transition testing is done.(Nov 21)                                                                                                                                                                                                                                                                                                                                               |
| 3.  | What is control flow graph? Explain how it is used in white box test design with an example. (Apr 22)                                                                                                                                                                                                                                                                                                                                                      |
| 4.  | Explain various white box test techniques with suitable test cases. Summarize the role of Oaths in white box testing and explain any two white box testing design. (Nov 16)                                                                                                                                                                                                                                                                                |
| 5.  | How white box testing methods related to TMM?                                                                                                                                                                                                                                                                                                                                                                                                              |
| 6.  | Explain in detail about Equivalence Class Partitioning. (Nov 16,17,18,Apr 17,18)                                                                                                                                                                                                                                                                                                                                                                           |
| 7.  | Explain State transition testing and error guessing. (Apr 18)                                                                                                                                                                                                                                                                                                                                                                                              |
| 8.  | How will you use white box approach to test case design?                                                                                                                                                                                                                                                                                                                                                                                                   |
| 9.  | Explain boundary value analysis method of black box testing with an example. (Nov 16,17,18,Apr 17,18)                                                                                                                                                                                                                                                                                                                                                      |
| 10. | Discuss in detail about code coverage testing. (Nov 16)                                                                                                                                                                                                                                                                                                                                                                                                    |
| 11. | Suppose you are testing defect coin problem artifacts, Identify the causes of various defects. What steps could have been taken to prevent the various classes of defects. (Apr 17)                                                                                                                                                                                                                                                                        |
| 12. | Explain the significance of control flow graph and Cyclomatic complexity in white box testing with a pseudo code for sum of 'n' numbers. (Nov 17)                                                                                                                                                                                                                                                                                                          |
| 13. | Discuss in detail about static testing and structural testing. Also write the difference between these testing concepts. (Nov 18,21,Apr 22)                                                                                                                                                                                                                                                                                                                |
| 14. | Outline the steps in constructing a control flow graph and computing Cyclomatic complexity with an example. (Apr 19)                                                                                                                                                                                                                                                                                                                                       |
| 15. | Discuss in detail about Boundary value analysis equivalence class portioning and state based testing with an example. (Apr 19)                                                                                                                                                                                                                                                                                                                             |
| 16. | Demonstrate the various black box test cases using Equivalence class partitioning and boundary values analysis to test a module for a "Banking system". State the functional requirements you are considering.(Nov 21)                                                                                                                                                                                                                                     |
| 17. | Mention at least four test case design strategies and explain them in detail.(Apr 21)                                                                                                                                                                                                                                                                                                                                                                      |
| 18. | Discuss below testing methods with examples. i) Compatibility Testing. ii) User documentation Testing. iii) Domain Testing. iv) Random Testing.(Apr 21)                                                                                                                                                                                                                                                                                                    |
| 19. | Develop black box test cases using equivalence class partitioning and boundary value analysis to test a module that is software component of an automated teller system. The module reads in the amount the user wishes to withdraw from his/ her account. The amount must be a multiple of \$5.00 and be less than or equal to \$200.00. Be sure to list any assumptions you make and label equivalence classes and boundary values that you use.(Apr 22) |

|     |                                                                                                                                                                                              |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20. | Discuss in detail the testing methodology which validates software system against functional requirements and specifications. Also explain the complete process with a neat diagram.(Apr 21) |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**UNIT III TEST DESIGN AND EXECUTION 6**

Test Objective Identification, Test Design Factors, Requirement identification, Testable Requirements, Modeling a Test Design Process, Modeling Test Results, Boundary Value Testing, Equivalence Class Testing, Path Testing, Data Flow Testing, Test Design Preparedness Metrics, Test Case Design Effectiveness, Model-Driven Test Design, Test Procedures, Test Case Organization and Tracking, Bug Reporting, Bug Life Cycle.

**UNIT-III/ PART-A**

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>What are the levels or major phases of testing?(Nov 18,Apr 22)</b><br>The major levels of Testing are as follows:<br>Unit testing (testing individual component), integration testing(testing integrated component), system testing( testing the entire system), and acceptance testing (testing the final system)                                                                                                                                                                                                       |
| 2. | <b>Define Unit Test. Give example. (Nov 17,21)</b><br>A unit is the smallest possible testable software component. It can be characterized in several ways. Forexample, a unit in a typical procedure-oriented software system:<br><ul style="list-style-type: none"> <li>❖ Performs a single cohesive function;</li> <li>❖ Can be compiled separately;</li> <li>❖ Is a task in a work breakdown structure (from the manager's point of view);</li> <li>❖ Contains code that can fit on a single page or screen.</li> </ul> |
| 3. | <b>What are the tasks to be performed in unit test?</b><br>To prepare for unit test the developer/tester must perform several tasks. These are:<br><ul style="list-style-type: none"> <li>❖ Plan the general approach to unit testing;</li> <li>❖ Design the test cases, and test procedures (these will be attached to the test plan);</li> <li>❖ Define relationships between the tests;</li> <li>❖ Prepare the auxiliary code necessary for unit test.</li> </ul>                                                        |
| 4. | <b>Define Test harness.(Apr 19)</b><br>The auxiliary code developed to support testing of units and components is called a test harness. The harness consists of drivers that call the target code and stubs that represent modules it calls.                                                                                                                                                                                                                                                                               |
| 5. | <b>What are the goals of Integration test?</b><br>Integration test for procedural code has two major goals:<br><ul style="list-style-type: none"> <li>❖ To detect defects that occur on the interfaces of units;</li> <li>❖ To assemble the individual units into working subsystems and finally a complete system thatis ready for system test.</li> </ul>                                                                                                                                                                 |
| 6. | <b>Define Cluster.</b><br>A cluster consists of classes that are related, for example, they may work together (cooperate) to support a required functionality for the complete system. A Cluster can be treated as a single entity or as a group of individual entities each having separate parameters and criteria.                                                                                                                                                                                                       |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7.  | <p><b>What are the items included in a Cluster Test Plan?</b></p> <p>Cluster Test Plan includes the following items:</p> <ul style="list-style-type: none"> <li>❖ Clusters this cluster is dependent on;</li> <li>❖ A natural language description of the functionality of the cluster to be tested;</li> <li>❖ List of classes in the cluster</li> <li>❖ A set of cluster test cases.</li> </ul>                                                                         |
| 8.  | <p><b>List the types of system test. (Nov 16)</b></p> <p>There are several types of system tests as follows:</p> <ul style="list-style-type: none"> <li>❖ Functional testing</li> <li>❖ Performance testing</li> <li>❖ Stress testing</li> <li>❖ Configuration testing</li> <li>❖ Security testing</li> <li>❖ Recovery Testing.</li> </ul>                                                                                                                                |
| 9.  | <p><b>Define Functional Testing.</b></p> <p>Functional tests are black box in nature. The focus is on the inputs and proper outputs for each function. Improper and illegal inputs must also be handled by the system. System behavior under the latter circumstances tests must be observed. All functions must be tested.</p>                                                                                                                                           |
| 10. | <p><b>Can you judge on the reason for system testing?</b></p> <ul style="list-style-type: none"> <li>❖ All types or classes of legal inputs must be accepted by the software.</li> <li>❖ All classes of illegal inputs must be rejected.</li> <li>❖ All possible classes of system output must exercise and examined.</li> <li>❖ All effective system states and state transitions must be exercised and examined.</li> <li>❖ All functions must be exercised.</li> </ul> |
| 11. | <p><b>What are two major types of requirements?</b></p> <p>Functional requirements. Users describe what functions the software should perform. We test for compliance of these requirements at the system level with the functional-based system tests.</p> <p>Quality requirements. There are nonfunctional in nature but describe quality levels expected for the Software. One example of a quality requirement is performance level.</p>                              |
| 12. | <p><b>What are the goals of system performance test?</b></p> <p>The goal of system performance tests is to see if the software meets the performance requirements. Testers also learn from performance test whether there are any hardware or software factors that impact on the system's performance. Performance testing allows testers to tune the system; that is, to optimize the allocation of system resources.</p>                                               |
| 13. | <p><b>Define Stress Testing. (Apr 19)</b></p> <p>When a system is tested with a load that causes it to allocate its resources in maximum amounts, this is called stress testing. For example, if an operating system is required to handle 10 interrupts/second and the load causes 20 interrupts/second, the system is being stressed.</p>                                                                                                                               |
| 14. | <p><b>What are the areas to be focused on during security testing?</b></p> <p>Password Checking, Password Expiration, Encryption, Legal and Illegal Entry with Passwords, Browsing, Trap Doors, Viruses</p>                                                                                                                                                                                                                                                               |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15. | <p><b>Define Configuration testing.</b></p> <p>Configuration testing allows developers/testers to evaluate system performance and availability when hardware exchanges and reconfigurations occur. Configuration testing also requires many resources including the multiple hardware devices used for the tests. If a system does not have specific requirements for device configuration changes then large-scale configuration testing is not essential.</p>                                                 |
| 16. | <p><b>What are the objectives of configuration testing according to Beizer?</b></p> <ul style="list-style-type: none"> <li>❖ Show that all the configuration changing commands and menus work properly.</li> <li>❖ Show that all interchangeable devices are really interchangeable, and that they each enter the proper states for the specified conditions.</li> <li>❖ Show that the systems' performance level is maintained when devices are interchanged, or when they fail.</li> </ul>                    |
| 17. | <p><b>What is security testing?</b></p> <p>Security testing evaluates system characteristics that relate to the availability, integrity, and confidentiality of system data and services. Users/clients should be encouraged to make sure their security needs are clearly known at requirements time, so that security issues can be addressed by designers and testers.</p>                                                                                                                                   |
| 18. | <p><b>Define Recovery testing.</b></p> <p>Recovery testing subjects a system to losses of resources in order to determine if it can recover properly from these losses. This type of testing is especially important for transaction systems, for example, on-line banking software.</p>                                                                                                                                                                                                                        |
| 19. | <p><b>What is Regression testing? (Nov 18)</b></p> <p>Regression testing is not a level of testing, but it is the retesting of software that occurs when changes are made to ensure that the new version of the software has retained the capabilities of the old version and that no new defects have been introduced due to the changes. Regression testing can occur at any level of test.</p>                                                                                                               |
| 20. | <p><b>Define a use case.</b></p> <p>A use case is a pattern, scenario, or exemplar of usage. It describes a typical interaction between the software system under development and a user.</p>                                                                                                                                                                                                                                                                                                                   |
| 21. | <p><b>Define Internationalization testing.</b></p> <p>Internationalization testing is the process of verifying the application under test to work uniformly across multiple regions and cultures. The main purpose of internationalization is to check if the code can handle all international support without breaking functionality that might cause data loss or data integrity issues. Globalization testing verifies if there is proper functionality of the product with any of the locale settings.</p> |
| 22. | <p><b>Based on what plan the scenario testing is done? (Nov 21)</b></p> <p>Scenario testing is a software testing activity that uses scenarios such as hypothetical stories to help the tester work through a complex problem or test system. The ideal scenario test is a credible, complex, compelling or motivating story the outcome of which is easy to evaluate.</p>                                                                                                                                      |
| 23. | <p><b>List the features of risk matrix test tool. (Apr 21)</b></p> <p>The risk matrix is based on two intersecting factors: the likelihood that the risk event will occur, and the potential impact that the risk event will have on the business. In other words, it's a tool that helps you visualize the probability the severity of a potential risk</p>                                                                                                                                                    |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24. | <p><b>Give the examples of security testing. (Apr 18)</b></p> <ul style="list-style-type: none"> <li>❖ Passwords</li> <li>❖ Encryption</li> <li>❖ Virus checkers</li> <li>❖ Detection and elimination of trap doors.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 25. | <p><b>What is localization testing? List its characteristics.</b></p> <p>Localization testing is performed to verify the quality of a product's localization for a particular target culture/locale and is executed only on the localized version of the product.</p> <p><b>Localization Testing - Characteristics:</b></p> <ul style="list-style-type: none"> <li>❖ Modules affected by localization, such as UI and content</li> <li>❖ Modules specific to Culture/locale-specific, language-specific, and region-specific</li> <li>❖ Critical Business Scenarios Testing</li> <li>❖ Installation and upgrading tests run in the localized environment</li> <li>❖ Plan application and hardware compatibility tests according to the product's target region.</li> </ul>                                                                                                                                                                                                                             |
| 26. | <p><b>Define acceptance testing.</b></p> <p>Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it has met the required criteria for delivery to end users.</p> <p>There are various forms of acceptance testing:</p> <ul style="list-style-type: none"> <li>❖ User acceptance Testing</li> <li>❖ Business acceptance Testing</li> <li>❖ Alpha Testing</li> <li>❖ Beta Testing</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                      |
| 27. | <p><b>Define alpha and beta testing. (Nov 16)</b></p> <p>Alpha testing takes place at the developer's site by the internal teams, before release to external customers. This testing is performed without the involvement of the development teams. Beta testing also known as user testing takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 28. | <p><b>Why is it important to design test harness for testing? (Apr 17,22, Nov 17)</b></p> <p>A test harness or automated test framework is a collection of software and test data configured to test a program unit by running it under varying conditions and monitoring its behavior and outputs. It has two main parts: the test execution engine and the test script repository.</p> <p>A test harness may provide some of the following benefits:</p> <ul style="list-style-type: none"> <li>❖ Increased productivity due to automation of the testing process.</li> <li>❖ Increased probability that regression testing will occur.</li> <li>❖ Increased quality of software components and application.</li> </ul> <p>Repeatability of subsequent test runs.</p> <ul style="list-style-type: none"> <li>❖ Offline testing (e.g. at times that the office is not staffed, like overnight).</li> <li>❖ Access to conditions and/or use cases that are otherwise difficult to simulate.</li> </ul> |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29. | <b>What is the advantage of bottom up integration? (Apr 18)</b><br>Bottom-up integration of the modules begins with testing the lowest level modules, those at the bottom of the structure chart. These are modules that do not call other modules. Drivers are needed to test these modules. The next step is to integrate modules on the next upper level of the structure chart whose subordinate modules have already been tested. After a module has been tested, its driver can be replaced by an actual module. |
| 30. | <b>Brief the importance of test plan in software testing.(Apr 21)</b><br>A test plan is the foundation of every testing effort. It helps set out how the software will be checked, what specifically will be tested, and who will be performing the test. By creating a clear test plan all team members can follow, everyone can work together effectively.                                                                                                                                                           |

### **UNIT-III / PART-B**

|     |                                                                                                                                                                                                                                                                              |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  | Discuss in detail about various levels of testing along with planning and designing test cases in each one of them.(Apr 21)                                                                                                                                                  |
| 2.  | State Unit test and describe about planning and designing of Unit test. (Apr 18,Nov 18)                                                                                                                                                                                      |
| 3.  | Summarize the issues that arise in class testing.                                                                                                                                                                                                                            |
| 4.  | (a) Why is it so important to design a test harness for reusability and show the approach you used for running the unit test and recording the results?<br>(b) Tabulate the key difference in integrating procedural oriented system as compared to object oriented systems. |
| 5.  | Elaborate in detail about the various types of system testing. (Nov 18)                                                                                                                                                                                                      |
| 6.  | Explain in detail about integration testing for procedures and functions with suitable diagrams. (Nov 16,17,21)                                                                                                                                                              |
| 7.  | Explain in detail about ad-hoc testing.                                                                                                                                                                                                                                      |
| 8.  | How would you identify the hardware and software for configuration testing and explain what testing techniques applied for website testing. (Nov 16,21,Apr 18)                                                                                                               |
| 9.  | Differentiate alpha testing from beta testing and discuss in detail about the phases in which alpha and beta testing is done. (Nov 17,Apr 22)                                                                                                                                |
| 10. | Explain the significance of Control flow graph and Cyclomatic complexity in white box testing with a pseudo code for sum of positive numbers. Also mention the independent paths with test cases. (Apr 16)                                                                   |
| 11. | With examples explain the following black box techniques to testing.(Apr 16)                                                                                                                                                                                                 |
| 12. | Write the importance of security testing and explain the consequences of security breaches, also write the various areas which have to be focused on during security testing. (Apr 18)                                                                                       |
| 13. | State the need for integration testing for procedural code. (Apr 18)                                                                                                                                                                                                         |
| 14. | Outline the importance of system testing with an example. What is regression testing? Outline the issues to be addressed for developing test cases to perform regression testing?(Apr 19)                                                                                    |
| 15. | Present an outline of testing object oriented systems.(Apr 19)                                                                                                                                                                                                               |
| 16. | Write a short note on below testing methods :<br>i. Alpha Testing, ii. Beta Testing, iii. OO Testing, iv. Website Testing.(Apr 21)                                                                                                                                           |
| 17. | Explain in detail about the testing strategy that ensures the availability, integrity, and confidentiality of system data and services. (Apr 22)                                                                                                                             |

|     |                                                                                                                                                      |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18. | Differentiate end to end testing and functional testing. When to apply end to end testing and detail various methods of end to end testing. (Apr 21) |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------|

**UNIT IV ADVANCED TESTING CONCEPTS**

Performance Testing: Load Testing, Stress Testing, Volume Testing, Fail-Over Testing, Recovery Testing, Configuration Testing, Compatibility Testing, Usability Testing, Testing the Documentation, Security testing, Testing in the Agile Environment, Testing Web and Mobile Applications

**UNIT-IV / PART-A**

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>Write about goal and its types?</b><br>A goal can be described as (i) a statement of intent, or (ii) a statement of a accomplishment that an individual or an organization wants to achieve.<br>❖ Business goal: to increase market share 10% in the next 2 years in the area of financial software.<br>❖ Technical goal: to reduce defects by 2% per year over the next 3 years.<br>❖ Business/technical goal: to reduce hotline calls by 5% over the next 2 years.<br>❖ Political goal: to increase the number of women and minorities in high management positions by 15% in the next 3 years. |
| 2. | <b>Write about policy, plan and milestone.</b><br>A <b>policy</b> can be defined as a high-level statement of principle or course of action that is used to govern a set of activities in an organization.<br>A <b>plan</b> is a document that provides a framework or approach for achieving a set of goals.<br><b>Milestones</b> are tangible events that are expected to occur at a certain time in the project's lifetime. Managers use them to determine project status.                                                                                                                        |
| 3. | <b>What are the high-level items included by a planner?</b><br>The planner usually includes the following essential high-level items.<br>❖ Overall test objectives.<br>❖ What to test (scope of the tests).<br>❖ Who will test<br>❖ How to test.<br>❖ When to test.<br>❖ When to stop testing.                                                                                                                                                                                                                                                                                                       |
| 4. | <b>Name the test Plan Components.</b><br>❖ Test plan identifier<br>❖ Introduction<br>❖ Items to be tested<br>❖ Features to be tested<br>❖ Approach                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 5. | <b>Define a Work Breakdown Structure. (Apr 18, Nov 21)</b><br>❖ A Work Breakdown Structure is a hierarchical or treelike representation of all the tasks that are required to complete a project.<br>❖ The WBS is used by project managers for defining the tasks and work packages needed for project planning                                                                                                                                                                                                                                                                                      |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6.  | <p><b>What is COCOMO Model? What is the use of cost driver?</b></p> <p>The test planner can use the COCOMO model to estimate total project costs, and then allocate a fraction of those costs for test. Application of the COCOMO model is based on a group of project constants that depend on the nature of the project and items known as cost drivers.</p>                                                                                                                                                                                                                                                                                                                                          |
| 7.  | <p><b>Analyze about Test Design Specification and its uses.</b></p> <ul style="list-style-type: none"> <li>❖ A test design specification is a test deliverable that specifies the requirements of the test approach .</li> <li>❖ It is used to identify the features covered by this design and associated tests for the features.</li> <li>❖ The test design specification also has links to the associated test cases and test procedures needed to test the features, and also describes in detail pass/fail criteria for the features</li> <li>❖ The test design specification helps to organize the tests and provides the connection to the actual test inputs/outputs and test steps.</li> </ul> |
| 8.  | <p><b>What are Test case Specifications?</b></p> <ul style="list-style-type: none"> <li>❖ Test Case Specification Identifier</li> <li>❖ Test Items</li> <li>❖ Input Specifications</li> <li>❖ Output Specifications</li> <li>❖ Special Environmental Needs</li> <li>❖ Special Procedural Requirements</li> <li>❖ Intercase Dependencies</li> </ul>                                                                                                                                                                                                                                                                                                                                                      |
| 9.  | <p><b>What is Test Summary Report?</b></p> <ul style="list-style-type: none"> <li>❖ Test Summary Report identifier</li> <li>❖ Variances</li> <li>❖ Comprehensiveness assessment</li> <li>❖ Summary of results</li> <li>❖ Evaluation</li> <li>❖ Summary of activities</li> <li>❖ Approvals</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                    |
| 10. | <p><b>What is the use of V-model in testing?</b></p> <p>The V-model is the model that illustrates how the testing activities can be integrated into each phase of the standard software life cycle.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 11. | <p><b>What is the role of Test Engineer?</b></p> <p>The test engineers design, develop, and execute tests, develop test harnesses, and set up test laboratories and environments. They also give input to test planning and support maintenance of the test and defect repositories.</p>                                                                                                                                                                                                                                                                                                                                                                                                                |
| 12. | <p><b>Differentiate decision and condition coverage. (Apr 16)</b></p> <p><b>Decision coverage</b> - It checks whether every edge of the program is covered or not i.e. each edge of every control structure (i.e. IF and CASE statements) are covered or not.</p> <p><b>Condition coverage</b> - It checks whether every conditional statements are covered or not. It evaluates for only true and false condition for each conditional statement not every edge of those conditional statements.</p>                                                                                                                                                                                                   |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13. | <p><b>What are the steps in forming the test group?</b></p> <ul style="list-style-type: none"> <li>❖ Upper management support for test function</li> <li>❖ Establish test group organization</li> <li>❖ Define education and skill levels</li> <li>❖ Develop job description</li> <li>❖ Interview candidates</li> <li>❖ Select test group members</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                |
| 14. | <p><b>What are the responsibilities for the developers/testers?</b></p> <ul style="list-style-type: none"> <li>❖ Working with management to develop testing and debugging policies and goals.</li> <li>❖ Participating in the teams that oversee policy compliance and change management.</li> <li>❖ Familiarizing themselves with the approved set of testing/debugging goals and policies, keeping up-to-date with revisions, and making suggestions for changes when appropriate.</li> <li>❖ When developing test plans, setting testing goals for each project at each level of test that reflect organizational testing goals and policies.</li> <li>❖ Carrying out testing activities that are in compliance with organizational policies.</li> </ul> |
| 15. | <p><b>List the Personal and Managerial Skills of a Test Lead</b></p> <ul style="list-style-type: none"> <li>❖ Organizational, and planning skills</li> <li>❖ Track and pay attention to detail</li> <li>❖ Determination to discover and solve problems</li> <li>❖ Work with others, resolve conflicts</li> <li>❖ Mentor and train others</li> <li>❖ Work with users/clients</li> <li>❖ Written/oral communication skills</li> </ul>                                                                                                                                                                                                                                                                                                                         |
| 16. | <p><b>List some of the Technical Skills.</b></p> <ul style="list-style-type: none"> <li>❖ General software engineering principles and practices</li> <li>❖ Understanding of testing principles and practices</li> <li>❖ Understanding of basic testing strategies, and methods</li> <li>❖ Ability to plan, design, and execute test cases</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                        |
| 17. | <p><b>What is the role of Test Lead?</b></p> <p>The test lead assists the test manager and works with a team of test engineers on individual projects. He or she may be responsible for duties such as test planning, staff supervision, and status reporting. The test lead also participates in test design, test execution and reporting, technical reviews, customer interaction, and tool training.</p>                                                                                                                                                                                                                                                                                                                                                |
| 18. | <p><b>What is the role of the Junior Test Engineer?</b></p> <p>The Junior test engineers are usually new hires. They gain experience by participating in test design, test execution, and test harness development. They may also be asked to review user manuals and user help facilities defect and maintain the test and defect repositories.</p>                                                                                                                                                                                                                                                                                                                                                                                                        |
| 19. | <p><b>List the test team hierarchy.</b></p> <ul style="list-style-type: none"> <li>❖ The test manager</li> <li>❖ The test lead</li> <li>❖ The test engineer</li> <li>❖ The junior test engineer.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20. | <p><b>Give some of the knowledge that must be shown by Candidates for Certification.</b></p> <p>Candidates for certification must show knowledge in areas that include:</p> <ul style="list-style-type: none"> <li>❖ Quality management;</li> <li>❖ Project management;</li> <li>❖ Measurement;</li> <li>❖ Testing;</li> <li>❖ Audits;</li> <li>❖ Configuration management.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 21. | <p><b>Define suspension and resumption criteria.(Apr 22)</b></p> <p>In the simplest of cases testing is suspended at the end of a working day and resumed the following morning. For some test items this condition may not apply and additional details need to be provided by the test planner. The test plan should also specify conditions to suspend testing based on the effects or criticality level of the failures/defects observed. Conditions for resuming the test after there has been a suspension should also be specified. For some test items resumption may require certain tests to be repeated.</p>                                                                                                                                                                                      |
| 22. | <p><b>Discuss on the role of manager in a test group. (Nov 16)</b></p> <p>Responsibilities of a manager are listed below:</p> <ol style="list-style-type: none"> <li>i. Planning the test strategy at a product or organizational level</li> <li>ii. Allocation of resources across the board</li> <li>iii. Implementing organizational policies</li> <li>iv. Effective management of meetings</li> <li>v. Keep everyone in sync by effective communication</li> <li>vi. Risk management</li> </ol>                                                                                                                                                                                                                                                                                                          |
| 23. | <p><b>What are the issues in testing object orient systems? (Nov 16)</b></p> <p>The issues in testing object orient systems are: Unit testing ,Composition , Encapsulation, Inheritance ,Polymorphism</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 24. | <p><b>State the limitations of statement coverage. (Apr 16)</b></p> <ul style="list-style-type: none"> <li>❖ It cannot test the false conditions.</li> <li>❖ It does not report that whether the loop reaches its termination condition.</li> <li>❖ It does not understand the logical operators.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 25. | <p><b>Define the term pass/Fail Criteria.</b></p> <p>Given a test item and a test case, the tester must have a set of criteria to decide on whether the test hasbeen passed or failed upon execution.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 26. | <p><b>List the various skills needed by a test specialist (Nov 17, Apr 22).</b></p> <p><b>Personal and Managerial level :</b> Organizational and planning skills and the ability to keep track of, and pay attention to, details. The determination to discover and solve problems, The ability to work with others and be able to resolve conflicts</p> <p><b>Technical level</b></p> <ul style="list-style-type: none"> <li>❖ An education that includes an understanding of general software engineering principles, practices, and methodologies</li> <li>❖ Strong coding skills and an understanding of code structure and behavior</li> <li>❖ A good understanding of testing principles and practices</li> <li>❖ A good understanding of basic testing strategies, methods, and techniques</li> </ul> |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27. | <p><b>What is the role of Test Summary Report? (Nov 17)</b></p> <p>Test summary report is a document which contains summary of test activities and final test results. After the testing cycle it is very important that to communicate the test results and findings to the project stakeholders so that decisions can be made for the software release.</p>                                                                                                                                                                                                                                                                                                         |
| 28. | <p><b>Mention the duties of component wise testing teams.(Nov 18)</b></p> <ul style="list-style-type: none"> <li>❖ Transmitting the software-under-test;</li> <li>❖ Developing test design specifications, and test cases;</li> <li>❖ Executing the tests and recording results;</li> <li>❖ Tracking and monitoring the test efforts;</li> <li>❖ Checking results;</li> <li>❖ Interacting with developers;</li> <li>❖ Managing and providing equipment;</li> <li>❖ Developing the test harnesses;</li> </ul>                                                                                                                                                          |
| 29. | <p><b>What is the need of Test Incident report? (Nov 18)</b></p> <p>The tester should record in a test incident report (sometimes called a problem report) any event that occurs during the execution of the tests that is unexpected, unexplainable, and that requires a follow-up investigation. It includes Test Incident Report identifier, Summary, Incident description and Impact.</p>                                                                                                                                                                                                                                                                         |
| 30. | <p><b>What is the function of Test Item Transmittal Report (TITR) or Locating test items? (Apr 18)</b></p> <p>TITR is not a component of the test plan, but is necessary to locate and track the items that are submitted for test. Each Test Item Transmittal Report has a unique identifier. It should contain the following information for each item that is tracked</p> <p>Version/revision number of the item, Location of the item , Persons responsible for the item (e.g., the developer), References to item documentation and the test plan it is related to, Status of the item, Approvals—space for signatures of staff who approve the transmittal.</p> |
| 31. | <p><b>Outline the need for a Test plan.(Apr 19, Nov 21)</b></p> <p>A test plan is also a systematic approach so it provides a better functional coverage. It also enables the managers to accurately estimate the required effort and cost to execute the testing process. Above all, it enables the test manager to control and track the progress using the test plan</p>                                                                                                                                                                                                                                                                                           |
| 32. | <p><b>What is a Test Log? (Apr 19)</b></p> <ul style="list-style-type: none"> <li>❖ Test log is one of the crucial test artifact prepared during the process of testing. It provides a detailed summary of the overall test run and indicates the passed and failed tests.</li> <li>❖ Additionally, test log also contains details and information about various test operations, including the source of issues and the reasons for failed operations.</li> </ul>                                                                                                                                                                                                    |
| 33. | <p><b>Give short note on cost estimation. (Apr 21)</b></p> <ul style="list-style-type: none"> <li>❖ Cost estimation in project management is the process of forecasting the financial and other resources needed to complete a project within a defined scope.</li> <li>❖ Cost estimation accounts for each element required for the project from materials to labor and calculates a total amount that determines a project's budget</li> </ul>                                                                                                                                                                                                                      |

|     |                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 34. | <b>Mention few aspects of software program which can be validated by performance testing. (Apr 21)</b> <ul style="list-style-type: none"> <li>❖ Speed – Determines whether the application responds quickly.</li> <li>❖ Scalability – Determines maximum user load the software application can handle.</li> <li>❖ Stability – Determines if the application is stable under varying loads</li> </ul> |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**UNIT-IV / PART-B**

|     |                                                                                                                                                                                                                                                                            |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  | Explain about Test plan components in detail. ( <i>Nov 16,17, Apr 18</i> )                                                                                                                                                                                                 |
| 2.  | Discuss on test plan attachments.                                                                                                                                                                                                                                          |
| 3.  | How will you report the test result? Explain in detail. (or) Name the reports of test results and the contents available in each test reports. ( <i>Apr 18</i> )                                                                                                           |
| 4.  | Discuss on the different skills needed by the test specialist. List and explain the skills needed by a test specialist. ( <i>Nov 16,17, Apr 18,22</i> )                                                                                                                    |
| 5.  | Explain about people and organizational issues in testing.                                                                                                                                                                                                                 |
| 6.  | Explain the concepts of test planning in detail. Also mention a way of defining a test plan. ( <i>Nov 18</i> )                                                                                                                                                             |
| 7.  | How will you build a test group? Explain the steps involved in forming a testing group. (or) Explain the concepts of building a test group. ( <i>Nov 17, 18</i> )                                                                                                          |
| 8.  | Explain in detail about the organization structures of testing teams in single product and multi product companies along with their advantages. ( <i>Nov 16,Apr 18,21</i> )                                                                                                |
| 9.  | Compare and contrast the role of debugging goals and policies in testing. ( <i>Nov 16</i> )                                                                                                                                                                                |
| 10. | How data flow testing aid in identifying defects in variable declaration and its use? ( <i>Apr 17</i> )                                                                                                                                                                    |
| 11. | Explain mutation testing with an example. ( <i>Apr 17</i> )                                                                                                                                                                                                                |
| 12. | Explain Weyuker's eleven axioms that allow testers to evaluate test adequacy criteria. ( <i>Apr 17</i> )                                                                                                                                                                   |
| 13. | Discuss the advantages and disadvantages of having an independent test group that is one that is a separate organizational entity with its own reporting structure. Why is it so important to integrate testing activities into the software life cycle? ( <i>Apr 19</i> ) |
| 14. | Explain the following test related documents and its components. ( <i>Apr 17</i> ) <ul style="list-style-type: none"> <li>(i)Test case specification</li> <li>(ii)Test Incident Report</li> </ul>                                                                          |
| 15. | Elaborate test management based on standards infrastructure, people, and product. ( <i>Nov 21</i> )                                                                                                                                                                        |
| 16. | What is the role of groups in policy development and test reporting? Give example. ( <i>Nov 21</i> )                                                                                                                                                                       |
| 17. | Elaborate different tools and techniques used for testing a software. ( <i>Apr 21</i> )                                                                                                                                                                                    |
| 18. | Compare and contrast the responsibilities of test engineer, senior test engineer, test lead and test manager in detail. ( <i>Apr 22</i> )                                                                                                                                  |

**UNIT V TEST AUTOMATION AND TOOLS**

Automated Software Testing, Automate Testing of Web Applications, Selenium: Introducing Web Driver and Web Elements, Locating Web Elements, Actions on Web Elements, Different Web Drivers, Understanding Web Driver Events, Testing: Understanding Testing.xml, Adding Classes, Packages, Methods to Test, Test Reports

**UNIT-V / PART-A**

|    |                                                                                                                                                                                                                                           |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <b>Define test data generator.</b><br>Automation should have scripts that produce test data to maximize coverage of permutations and combinations of inputs and expected output or result comparison. They are called test data generator |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2.  | <b>What is Test Automation?(Nov 21)</b><br>In software testing, test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. Automation saves times as software can execute test cases faster than human do test automation can free the test engineers from mundane tasks and to focus on creative task. Automated tests can be more reliable. Automation helps in immediate testing                                                                                                               |
| 3.  | <b>What are the different generations of automation?</b><br>First Generation – Record and playback, Second Generation – Data Driven, Third Generation – Action driven                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 4.  | <b>What is configuration file?</b><br>A configuration file contains a set of variables that are used in automation. The variables could be for the test framework or for other modules in automation such as tools and metrics or for the test suite or for a set of test cases or for a particular test case. A configuration file is important for running the test cases for various execution conditions and for running test for various input and output conditions. The values of these variables in this configuration file can be changed dynamically to achieve different execution input, output and state condition. |
| 5.  | <b>Differentiate effort variance and schedule variance.</b><br>Effort variance provides a quantitative measure of the relative difference between the revised and actual efforts. Schedule variance is the deviation of the actual schedule from the estimated schedule.                                                                                                                                                                                                                                                                                                                                                         |
| 6.  | <b>Define test framework.(Apr 22)</b><br>A testing framework or more specifically a testing automation framework is an execution environment for automated tests. It is the overall system in which the tests will be automated. It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing. The Testing framework is responsible for: Defining the format in which to express expectations, Creating a mechanism to hook into or drive the application under test, Executing the tests                                                                   |
| 7.  | <b>List the application areas of text mining. (Apr 17)</b><br>Cybercrime prevention, Customer care service, Fraud detection, Contextual Advertising                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 8.  | <b>Explain the work of report generator.</b> <ul style="list-style-type: none"> <li>❖ Once the test run result is available, the next step is to prepare the test report and metrics. Preparing report is a complex and time consuming effort and hence it should be part of the automation design.</li> <li>❖ The module that takes input and prepares a formatted report is a called report generator.</li> <li>❖ The periodicity of the report is different such as daily, weekly, monthly or milestones report.</li> </ul>                                                                                                   |
| 9.  | <b>What are the criteria for selecting test tools?</b><br>Meeting requirements, Technology expectation, Training/skills, Management aspects and cost                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 10. | <b>What are the different software automation testing tools?(Apr 19)</b><br>Rational functional tester, Robot framework, Silk Test, Selenium, Cucumber, Appium                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11. | <p><b>Define instrumented code.</b></p> <p>Test tool requires their libraries to be linked with product binaries. When the libraries are linked with the source code of the product, it is called instrumented code.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 12. | <p><b>List down the basic concepts of extreme programming.</b></p> <ul style="list-style-type: none"> <li>❖ Unit test cases are developed before the coding phase starts.</li> <li>❖ Code is written for test cases and are written to ensure test cases pass</li> <li>❖ All the unit tests must run 100% all the time</li> <li>❖ Everyone owns the product; they often cross boundaries.</li> </ul>                                                                                                                                                                                                                                                                        |
| 13. | <p><b>What is test defect metrics?</b></p> <ul style="list-style-type: none"> <li>❖ A set of metrics helps to understand how the defects that are found can be used to improve testing and product quality.</li> <li>❖ The defects are classified by defect priority and defect severity.</li> <li>❖ Defect priority provides management perspective for the order of defect fixes.</li> <li>❖ The severity of defects provides the test team a perspective of the impact of that defect in product functionality.</li> </ul>                                                                                                                                               |
| 14. | <p><b>List out the different types of metrics.</b></p> <p>Project metrics: A set of metrics that indicate how the project is planned and executed.<br/>     Progress metrics: A set of metrics that tracks how the different activities of the project are progressing.<br/>     Productivity metrics: A set of metrics that helps in planning and estimating of testing activities.</p>                                                                                                                                                                                                                                                                                    |
| 15. | <p><b>What is metrics program and list down the steps in metrics program?</b></p> <p>Metrics drive information from raw data with the view to help in decision making. Steps:<br/>     Identify what to measure, Transform measurement to metrics, Decide operational requirements, Perform metrics analysis, Take actions and follow up and Refine.</p>                                                                                                                                                                                                                                                                                                                    |
| 16. | <p><b>What are the types of reports?</b></p> <p>Executive report: gives a very level status, Technical report: moderate level of details of test run, Detailed or debug report: to debug the failed test case.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17. | <p><b>What do you mean by closed defect distribution?</b></p> <p>The objective of testing is not only to find defects. The testing team also has the objective to ensure that all defects found through testing are fixed so that the customer gets the benefit of testing and the product quality improves. To ensure the most of the defects are fixed, the testing team has to track the defects and analyze how they are closed. The closed defect distribution helps in this analysis.</p>                                                                                                                                                                             |
| 18. | <p><b>Define release metrics.</b></p> <p>The decision to release a product would need to consider several perspectives and several metrics. The release metrics provides the guidelines in making the release decision. Some release metrics are test case executed, effort distribution, defect find rate, defect fix rate, outstanding defect trends, priority outstanding defect trends, weighted defect trends, defect density and defect removal rate, age analysis of outstanding defects, introduces and reopened defects, defects per 100 hours of testing, test cases executed for 100 hours of testing, test phase effectiveness, closed defect distribution.</p> |
| 19. | <p><b>How the defect metrics can be used in improving the development activities?</b></p> <p>Component wise defect distribution, Defect density and defect removal rate and Age analysis of outstanding defects Introduced</p>                                                                                                                                                                                                                                                                                                                                                                                                                                              |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20. | <p><b>What are the uses of walkthrough? (or) What is Walk through? (Nov 17,Apr 18,22)</b></p> <p>In software engineering, a walkthrough or walk-through is a form of software peer review "in which a designer or programmer leads members of the development team and other interested parties go through a software product, and the participants ask questions and make comments about possible errors, violation of development standards, and other problem. Uses of Walkthrough are Improves the product overall, Detect the hidden and unseen bugs, Evaluate the test cases and documents, Understanding of the review object, Discuss the alternative for design, Discuss the alternative for implementations.</p> |
| 21. | <p><b>Give the formula for defects per 100 hours of testing</b></p> <p>Defect per 100 hours of testing normalizes the number of defects found in the product with respect to the effort spent.</p> <p>Defect per 100 hours of testing=(Total defect found in the product for a period/Total hours spent to get those defects)*100</p>                                                                                                                                                                                                                                                                                                                                                                                      |
| 22. | <p><b>What is the different purpose of productivity metrics?</b></p> <p>Estimating for the new release, Finding out how well the team is progressing, understanding the reason for variation in result, Estimating the number of defects that can be found, Estimating release date and quality, Estimating the cost involved in the release.</p>                                                                                                                                                                                                                                                                                                                                                                          |
| 23. | <p><b>What are the different test defect metrics?</b></p> <p>Defect find rate, Defect fix rate, outstanding defects rate, Priority outstanding rate, weighted defects trend, Defect cause distribution.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 24. | <p><b>Mention the criteria for selecting test tool. (Nov 16)</b></p> <p>Meeting requirements, Technology expectations, Training, Management aspects, Cost, Complexity</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 25. | <p><b>Mention the challenges in automation. (Nov 17, Apr 17,21)</b></p> <p>Effective Communicating and Collaborating in Team, Selecting a Right Tool, Demanding Skilled Resources, Selecting a Proper Testing Approach, Cost involved, Scope of the Investment.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 26. | <p><b>Distinguish between mile stone and deliverable. (Nov 16)</b></p> <p>A milestone and a deliverable both mark a point in the project where a "substantial" achievement is done. A milestone marks an event in time in your schedule where something is accomplished, a deliverable is actual work completed (delivered) at that point in time.</p>                                                                                                                                                                                                                                                                                                                                                                     |
| 27. | <p><b>Mention the types of testing amenable to automation. (Apr 17).</b></p> <p>Stress testing, Reliability testing, Scalability testing, Performance testing.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 28. | <p><b>What is the need for automated testing? (Nov 18)</b></p> <p>Automation saves time as software can execute test cases faster than humans do. The time saved can be used effectively by test engineers to i)Develop additional test cases to achieve better coverage ii) Perform some esoteric or specialized tests like adhoc testing.</p>                                                                                                                                                                                                                                                                                                                                                                            |
| 29. | <p><b>What are the goals of reviewers?(Apr 18)</b></p> <p>Goals of reviewers are to find and remove errors early in the software development life cycle. Reviews of reviewers are used to verify documents such as requirements, system designs, code, test plans and test cases.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                      |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30. | <p><b>Define progress metrics. (Nov 18)</b></p> <p>Progress metrics is a type of metrics that tracks how the different activities of the project are progressing.</p> <p>Example:</p> <ul style="list-style-type: none"> <li>❖ Man-hours/test case executed</li> <li>❖ Planned hours/actual hours</li> <li>❖ Test cases executed/planned</li> <li>❖ Test cases executed/defects found</li> <li>❖ Resource needed/Resource used</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 31. | <p><b>Mention two methods by which you can do production analytics</b></p> <p>MTBF (mean time between failures) is a measure of how reliable a hardware product or component is.</p> <ul style="list-style-type: none"> <li>❖ For most components, the measure is typically in thousands or even tens of thousands of hours between failures.</li> <li>❖ The formula for mean time between failure or MTBF is: <math>T/R</math>, where <math>T</math> is the total number of unit hours from the trial in question, and <math>R</math> is the number of failures.</li> <li>❖ Mean time to repair (MTTR) is a basic measure of the maintainability of repairable items. It represents the average time required to repair a failed component or device.</li> <li>❖ To calculate MTTR, divide the total maintenance time by the total number of maintenance actions over a given period of time</li> </ul> |
| 32. | <p><b>What do you mean by Test Metrics?(Nov 21)</b></p> <ul style="list-style-type: none"> <li>❖ Software Testing Metric is defined as a quantitative measure that helps to estimate the progress, quality, and health of a software testing effort.</li> <li>❖ A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 33. | <p><b>List various productivity metrics in software testing. (Apr 21)</b></p> <ul style="list-style-type: none"> <li>❖ Test Case Productive Preparation = Total test steps / effort (hours)</li> <li>❖ Test Execution Summary</li> <li>❖ Test Case Coverage</li> <li>❖ Defect Acceptance</li> <li>❖ Defect Rejection</li> <li>❖ Test Efficiency</li> <li>❖ Effort Variance</li> <li>❖ Schedule Variance</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 34. | <p><b>Outline the need for test metrics. (Apr 19)</b></p> <ul style="list-style-type: none"> <li>❖ Software Testing Metrics are useful for evaluating the health, quality, and progress of a software testing effort.</li> <li>❖ Without metrics, it would be almost impossible to quantify, explain, or demonstrate software quality.</li> <li>❖ Metrics also provide a quick insight into the status of software testing efforts, hence resulting in better control through smart decision making.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                          |

|     |                                            |                                                                                                                                                                |
|-----|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 35. | <b>Mention the Test metric life cycle.</b> |                                                                                                                                                                |
|     | <b>Metrics Life cycleStages</b>            | <b>Different stages of Metrics life cycle</b>                                                                                                                  |
|     | Analysis                                   | Identification of the Metrics, Define the identified QA Metrics                                                                                                |
|     | Communicate                                | Explain the need for metric to stakeholder and testing team<br>Educate the testing team about the data points to need to be captured for processing the metric |
|     | Evaluation                                 | Capture and verify the data, Calculating the metrics value using the data captured                                                                             |
|     | Report                                     | Develop the report with an effective conclusion, Distribute the report to the stakeholder and respective representative                                        |

### **UNIT-V / PART-B**

|     |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  | How do you calculate defect density and defect removal rate? Discuss ways to improve these rates for a better Quality product?                                                                                                                                                                                                                                                                                                |
| 2.  | Explain the various generations of automation and the required skills for each( <i>Apr 17</i> )<br>(or) Write short notes on classification of automation testing. ( <i>Nov 18,21</i> )                                                                                                                                                                                                                                       |
| 3.  | Write short notes on the scope of automation. ( <i>Nov 18</i> )                                                                                                                                                                                                                                                                                                                                                               |
| 4.  | Explain the design and architecture for automation. ( <i>Nov 16,17,Apr 22</i> )                                                                                                                                                                                                                                                                                                                                               |
| 5.  | Discuss the challenges in automation. ( <i>Nov 16</i> )                                                                                                                                                                                                                                                                                                                                                                       |
| 6.  | How to select the test tool and also discuss the criteria and steps in selecting the tool? (or)<br>Discuss in detail about selecting a test tool in test automation. ( <i>Nov 18</i> )                                                                                                                                                                                                                                        |
| 7.  | What are the steps involved in a metrics program. Briefly explain each step.                                                                                                                                                                                                                                                                                                                                                  |
| 8.  | Discuss on different project and progress metrics?                                                                                                                                                                                                                                                                                                                                                                            |
| 9.  | What are metrics and measurements? Illustrate the types of product metrics. Discuss various metrics and measurements in software testing. Explain various types of progress metrics. ( <i>Nov 16,17,Apr 17,18,21,22</i> )                                                                                                                                                                                                     |
| 10. | Discuss the type of review. Explain the various components of review plan. ( <i>Apr 18</i> )                                                                                                                                                                                                                                                                                                                                  |
| 11. | Developing software to test the software is called test automation. Test automation can help address several problems. Justify Draw the framework for test automation. ( <i>Apr 19</i> )                                                                                                                                                                                                                                      |
| 12. | Outline project, product and productivity metrics with relevant examples. ( <i>Apr 19,Nov 21</i> )                                                                                                                                                                                                                                                                                                                            |
| 13. | Assume you are working in an on-line fast food restaurant system. The system reads customer orders. Relays orders to the kitchen, calculates the customer's bill and give change. It also maintains inventory information. Each wait person has a terminal. Only authorized wait persons and a system administrator can access the system. Describe the tests that is suitable to the test the application. ( <i>Nov 21</i> ) |
| 14. | How do you calculate defect density and defect removal rate? Discuss ways to improve these rates for a better Quality product?                                                                                                                                                                                                                                                                                                |
| 15. | Explain the structure of test group with component wise testing team? ( <i>Apr 21</i> )                                                                                                                                                                                                                                                                                                                                       |