Manual Testing

1. **What is Testing?**

Software testing is the process of verifying and validating the functionality of the application to check if it satisfies client’s or customer’s requirements. It is the process of finding defects in an application.

1. **What are the types of software testing...?**

* **Manual Testing** - To test the any application/software without any automated tools that is called manual.
* **Automation Testing** - To test the any application without any manual efforts.

1. BA - Business Analyst. (Service Based Company)
2. PO - Process Owner. (Product Based Company)

Lets talk about Manual Testing first:

Manual testing is a software testing process in which test cases are executed manually without using any automated tool.

All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application.

Test case reports are also generated manually.

The difference between expected output and output, given by the software, is defined as a defect. The developer fixed the defects and handed it to the tester for retesting.

**Why we need Manual Testing:**

Whenever an application comes into the market, and it is unstable or having a bug or issues or creating a problem while end-users are using it.

If we don't want to face these kinds of problems, we need to perform one round of testing to make the application bug free and stable and deliver a quality product to the client, because if the application is bug free, the end-user will use the application more conveniently.

If the test engineer does manual testing, he/she can test the application as an end-user perspective and get more familiar with the product, which helps them to write the correct test cases of the application and give the quick feedback of the application.

**Types of Manual Testing:**

1. White Box Testing
2. Black Box Testing
3. Gray Box Testing

**White Box testing**

White box testing is used to test the internal structure of the application

The term 'white box' is used here because white box testing is done by the developer. As developers have programming skill they can understand the internal structure of the application.

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.

**White Box testing:**

The black box testing is done by the Test Engineer, where they can check the functionality of an application or the software according to the customer /client's needs. In this, the code is not visible while performing the testing; that's why it is known as black-box testing.

**First, we will read about black box:** This is kind of testing, in this method tester selects a function and gives input value to check its functionality and check if software is giving expected output or not. If the software gives expected output, then it is passed otherwise failed. Test team report the result to the development team and then test the next function.

Black box testing

**Steps of Black of Testing:**

1. The black box test is based on condition of requirements, so it is examined in the beginning
2. In the second steps tester creates a positive test scenario and vice versa test scenario by taking valid and invalid input values to check that software is processing correctly or not.
3. In third steps, the tester develops various test cases such as decision table, all pair test, equivalent division, error estimation, cause-effect graph etc.
4. The fourth phase contain the execution of all test cases.
5. In the fifth steps, the tester compares the expected output against the actual output.
6. In the sixth and final steps, if there is any error in software, then it is fixed and test again.

**Gray Box Testing:**

Grey Box Testing is a combination of black and white box testing because in that we have to perform the functionality of the application and internal structure of the application as well.

**What (Software Development Life Cycle)::**

Phase of SDLC in Agile:

1. Requirement gathering:
2. Designing phase
3. Implementation Phase
4. Testing Phase
5. Deployment/Delivered phase
6. Maintenance Phase

**What is STLC (Software testing Life Cycle)::**

Phase of STLC is:

1. Requirement Analysis
2. Test plan creation
3. Environment Setup
4. Test Case Execution
5. Defect Logging (process of finding defects)
6. Test Cycle closure

More understanding here:

The first step of the manual testing procedure is requirement analysis. In this phase, tester analyses requirement document 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.

In Test plan creation, define scope of the software and note down method involved in testing.

Then we create a test environment to test the application. The testing team is not involved in setting up the testing environment, its developers who create it.

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.

In Defect logging analysis, we find out defect and categorize in types and severity.

The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports. These are also used in future if there is a software with the same specification.

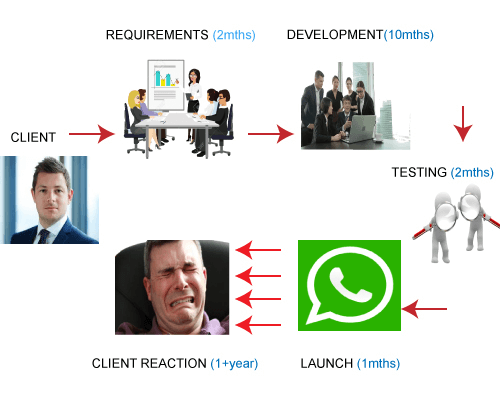
**Steps to perform Manual Testing:**

1. First, tester observes all documents related to software, to select testing areas.
2. Tester analyses requirement documents to cover all requirements stated by the customer.
3. Tester develops the test cases according to the requirement document.
4. All test cases are executed manually by using Black box testing and white box testing.
5. If bugs occurred then the testing team informs the development team.
6. The Development team fixes bugs and handed software to the testing team for a retest.

**What are 7 principal of software testing??**

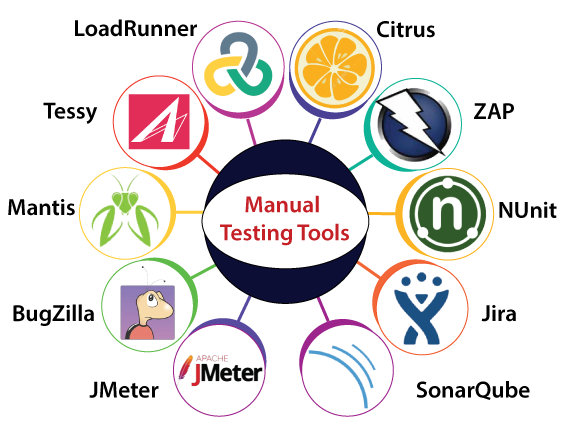
1. **Testing shows presence of defects**- (Testing can reduce the number of defects but not remove all defects)
2. **Exhaustive Testing is not possible**- (testing for every possible combination of input is not possible)
3. **Early Testing**
4. **Defect Clustering**- (most of the bug are linked to small module)
5. **Pesticide Paradox-** (don’t test same test case again and again without updating)
6. **Testing is context-dependent-** (should be test as per context or requirements)
7. **Absence of error fallacy- (**If your software or system is unusable (or does not fulfill users' wishes) then it does not matter how many defects are found and fixed – it is still unusable.)

**Example of Waterfall Modal:**



Suppose the client wants an app like a WhatsApp, so he reaches to the company where both the company and the client had a discussion for 2 months. The company made the documentation of all the requirements in 2 months. Now, the development team starts developing the software and suppose it took around 10 months to develop the software. It means that 12 months have been used, i.e., 2 months in requirement phase and 10 months in a development phase, but still the client does not have the idea about the internal phases. Once the development is completed, testing is done, and it will take around 2 months for software quality testing. Once the testing is done, it goes to the integration and launch so that WhatsApp will become live. However, when it reaches to the client, then the client says that it has taken more than a year and the software that I received was not what I expected. This happened because the client had only verbal communication with the software team. If the client wants some changes in the software, then the whole process will be executed again.

**Manual Testing Tools:**



The most regularly used bug tracking tool is **JIRA**, which is an open-source tool. It is used for bug tracking, project management, and issue tracking.

In this tool, we can easily track all kinds of bugs or defects related to the software and produced by the test engineers.

JIRA is based on the Agile methodology and the current version of the Jira is 6.

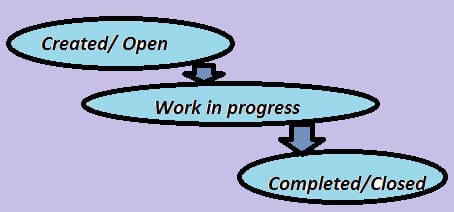
**What are the statuses of JIRA ...?**

1. Backlog
2. In dev
3. Dev done
4. In QA
5. QA Done
6. Ready for production (Bug)
7. Resolved (CR)
8. Done (Task)

**Explain JIRA workflow:**

JIRA workflow includes the creation of an issue, actions performed to fix the issue and the last step includes the closing or say completion of the issue after verification.

**Refer the below diagram for better understanding:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2016/10/JIRA-workflow.jpg)

**Steps to create an issue in JIRA:**

Whenever an issue or defect is encountered while testing, it needs to be reported so that the developers can work on it and take the necessary action to fix it.

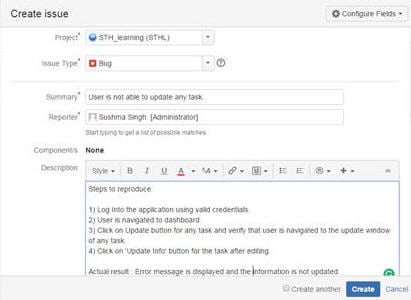
**We will see step by step as how an issue is created in Atlassian JIRA.**

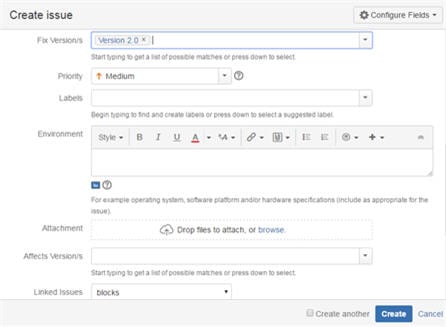
**a)** Log in to your JIRA account by using valid credentials and get directed to the dashboard.

**b)** Click on ‘Create’ button displayed and you will be navigated to a window for creating an issue.

**c)** Enter all the necessary details as required to create an issue. The below field example can be understood better by the image below.

* In the ***Project***field*, a*project for which we are creating an issue is selected. In this example: STH\_Learning(STHL) is selected from the dropdown containing all the available projects.
* In the ***Issue type*** field, the nature of the issue is selected from the dropdown which contains options like Bug, Task, Improvement, Story, New Feature, etc. In this example, ‘Bug’ is the nature of the issue.
* The***Summary*** field contains the one-line title of the issue which imparts the critical information about the issue in a summarized way. The more effective the issue headline, the more you can show the criticality of the issue. Of course, the headline should be easily understood without any chances of misinterpretation. The example I have taken here, however, is not much critical.
* The ***Reporter*** is the one who reports the issue. In most of the cases, the name of the Project Manager is selected in this field.
* In ***Description***field, the detailed description of the issue is written. As you can see in the below example screenshot, Steps to reproduce the issue, Actual result, Expected result are included in the description.
* In the ***Affect Version***field***,***the current build version of the project is selected in which the issue has been encountered.
* ***Fix version*** field is basically selected by the concerned developer people, who choose the version as and when their work for the particular issue has been finished and the issue has been fixed.
* ***Priority*** field defines which issue should be considered first to be fixed. Tester selects the priority of the issue from the dropdown based on its effect on the application. This example issue is basically of a Medium priority.
* In the ***Attachment*** field, any video or screenshot related to the issue is being uploaded.
* In***Environment*** field, operating system and browser details are mentioned on which issue has been encountered.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2016/10/Create-issue-1.jpg)

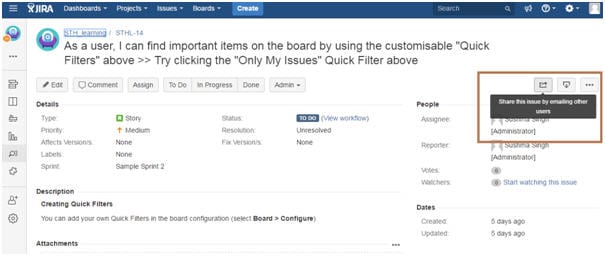
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2016/10/Create-issue-2.jpg)

**d)** After all the details have been completed, click on the ‘Create’ button displayed on the window to create the new issue.

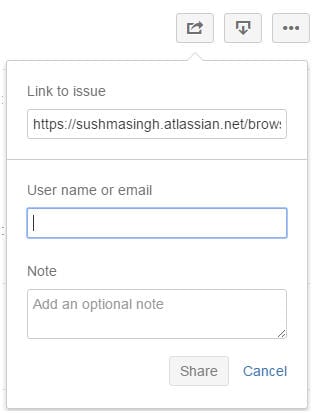
**e)** The issue id is generated which can be used in the future as a reference for tracking the progress of the issue;

**How to share an issue with another user in JIRA:**

An issue can be shared with other users in JIRA by using the share option available on the issue description page.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2016/10/share-option.jpg)

When share option for any particular issue is clicked, it contains the link to the issue to be shared along with ‘Username or Email’ and ‘Note’ field that has to be filled.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2016/10/share-option-1.jpg)

**8.What is user story…?**

User Stories is short requirement or request written from the perspective of end user.

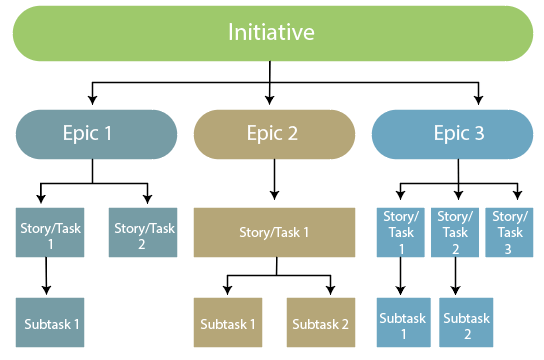
A user story is **an informal, general explanation of a software feature written from the perspective of the end user**. Its purpose is to articulate how a software feature will provide value to the customer

**What is EPIC in JIRA?**

* Jira epic is a large user story which is broken into smaller tasks (user stories) based on the customer or end-user needs.
* Based on the customer needs, the task is added or removed from the epic as necessary.

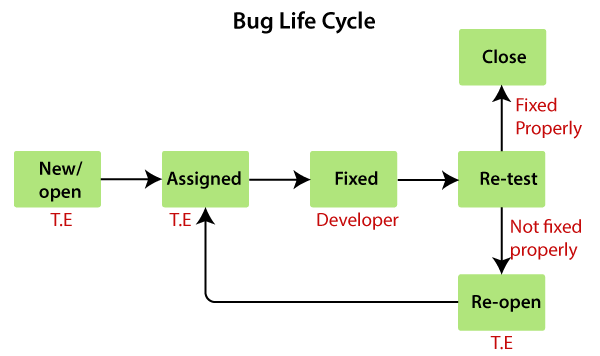
**What is Initiative?**

Initiative is collection of epics that leads to common goal.



**Bug Life Cycle OR JIRA Bug Life Cycle:**

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 analyse 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.

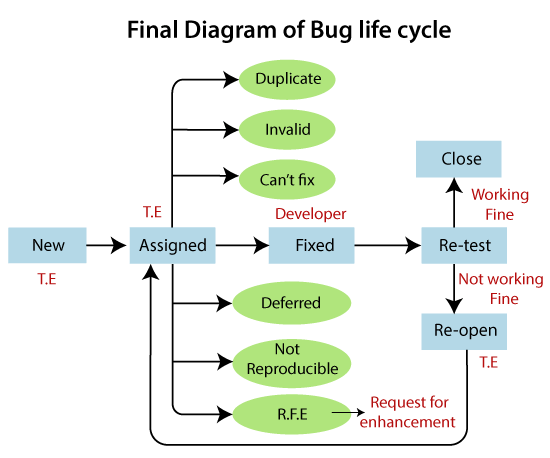
This process is going on continuously until all the bugs are fixed and closed.

**Final Diagram of Bug Life Cycle:**

JIRA bug life cycle is also known as a defect life cycle. The bug life cycle consists of a set of states that a bug goes through. The number of states that the bug goes through varies from project to project. We can define the bug as an error, flaw or we can say that when the actual output does not match with the expected output, it is known as bug or defect. Both the terms, i.e., bug and defect are commonly used but the most popular is a bug. A bug can be generated at any stage of the **SDLC**(Software Development Life Cycle), it could exist in requirements gathering, designing phase where SRS document is designed, development phase, testing phase or user acceptance testing done by the end-user at the time of using the application.

A bug has its life cycle from the point when the bug is logged in to the point the bug is closed. Bug undergoes the following states:

* New
* Assigned
* Open
* Fixed
* Retesting
* Reopen
* Verified
* Closed



**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

**Duplicate:**

When the same bug has been reported multiple times by the different test engineers are known as a **duplicate** bug.

**Not Reproducible:**

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

**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.

**Deferred/postpone**

The deferred/postpone is a status in which the bugs are postponed to the future release due to time constraints.

**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**.

**Entry and Exit criteria?**

* **Entry Criteria**: Entry Criteria are the criteria or task that must be completed before testing can begin.
* **Exit Criteria**: Exit Criteria defines the items that must be completed before testing can be finished.



**What are Test Cases?**

* **Test** case are group of conditions under which tester determine if a software is working as per the customer’s requirements.
* Test case include preconditions, case name, input condition and expected results.
* It is a detail documents that contains all possible input (positive as well as negative) and the navigation steps which are used for test navigation process.

**When do we write Test Case?**

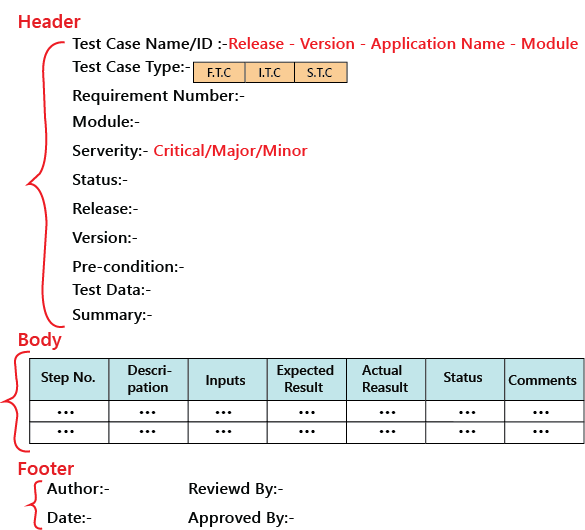
When developer start developing a project and say that they need few months to develop the project in the meantime, test team will start writing the test cases.

**Why do we write Test Cases?**

The purpose of test case is to determine, if different feature of a software is performing as expected and to confirm that software satisfies all the customer’s requirement.

**Test Cases Templates:::**

Below is a test case template, but it is not standard one as it may be different for different companies and even for different software.



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:**

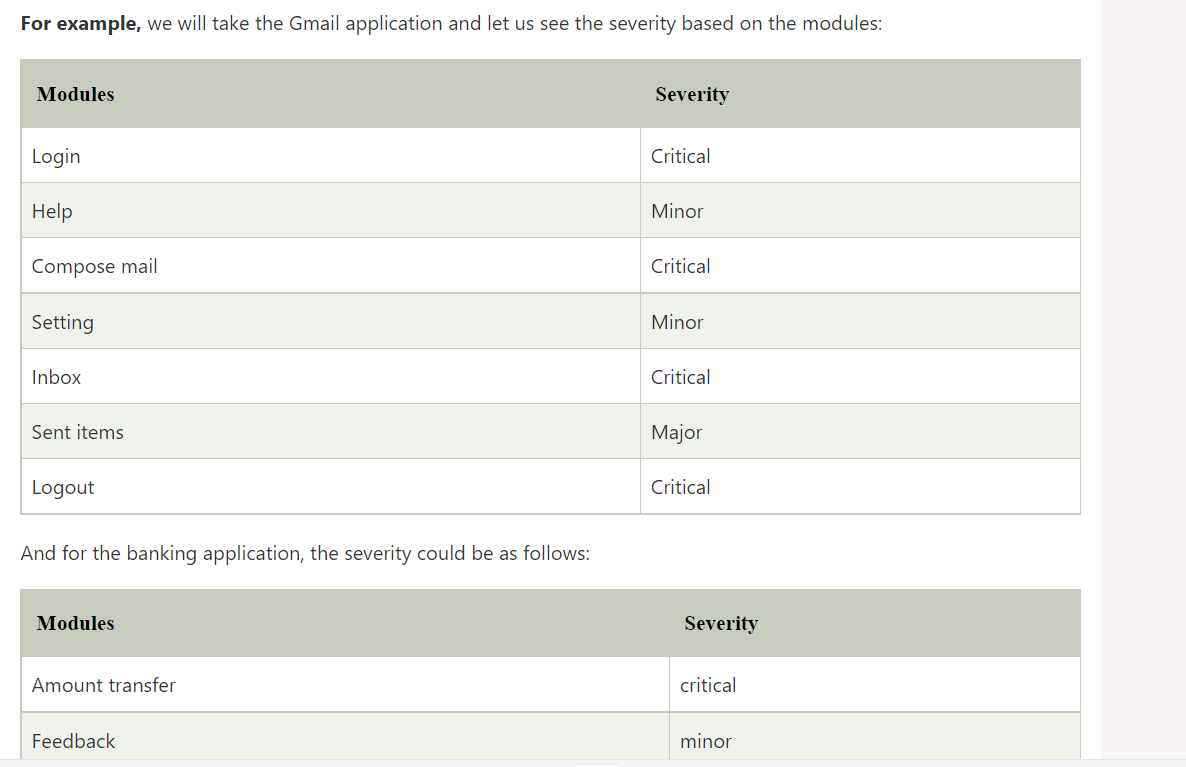
**I**t is the data configuration or the data setup that needs to be created for the testing. **For example**: In an application, we are writing test cases to add users, edit users, and delete users. The per-condition will be seen if user A is added before editing it and removing it.

* **Test Data:**

**T**hese 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 themself generate the username and password.

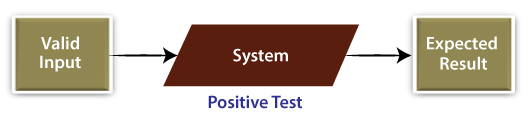
* **Severity:**

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 text execution process always depends on the severity of the test cases



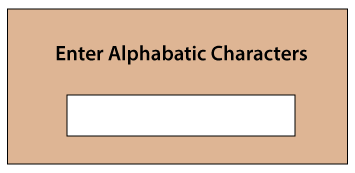
**Positive test cases::**

* When we test the application in happy flow with the positive intent.
* It validates how the application performs for the positive set of input. In this type of testing, we will enter the valid data set as the input value.



* We will implement positive testing to validate the exact working of different software modules with the lines of estimated performance in response to valid data input.

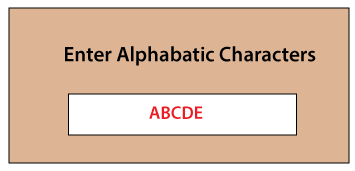
Example of positive test case: suppose we have one **Textbox** that can only accept **alphabetic character values.**



**Therefore, in this case, the positive test scenario is as follows:**

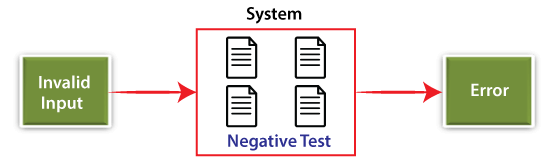
**Positive Test Scenario:**

* Enter the **alphabetical character values** and verify that the application receives the input values under test or not.



**What is Negative test case?**

* When we test the application to opposite of the functionality or behaviour of the application that is called negative test cases.
* Negative testing ensures that the plot of the application is according to the requirements and can handle the unwanted input and user behaviour
* we can say that negative testing is implemented to guarantee that the software product under test does NOT fail when an unpredicted input is given. **It is also known as failure testing** or **error path testing.**



**For example:** Suppose we have one sample form to enter the values for the name, phone number, and Pincode fields. In such case, the negative input could be the following:

**Name: 838383**

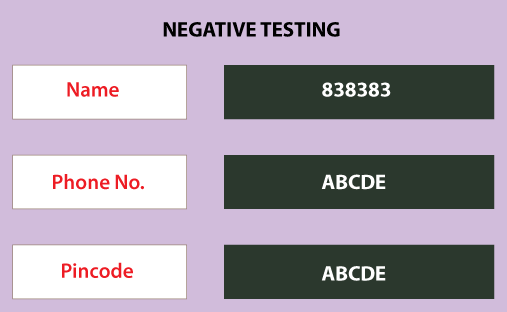
(As the name field only accepts the alphabetic character value)

**Phone No: ABCDE**

(As the Phone no field only accepts the he values of the number)

**Pincode: ABCDE**

(As the Pincode field only accepts the he values of the number)



**Test Case Review process**

While writing test cases, tester may miss some scenario, input etc that may affect the software testing process. To avoid this, once test case is written we do **review and approval process.**

So, all the test case are sent for the review, where reviewer will check the test cases. And if any mistake is found in test case reviewer will correct in and sent back to tester.

The reviewer opens the test case with the corresponding requirement and checks the **correctness of the test case, proper flow, and maximum test coverage**.

When the test cases are written, review, and approved, it will be stored in one centralized location, which is known as **Test Case Repository**.

**What is the difference between test case and test scenario?**

|  |  |
| --- | --- |
| **Test Cases** | **Test scenario** |
| **Test** case are group of conditions under which tester determine if a software is working as per the customer’s requirements. | Test scenario give one line information about what to test. |
| It is a detail documents that contains all possible input (positive as well as negative) and the navigation steps which are used for test navigation process | It is one liner statement and connected with test cases. |
| These are **low-level actions**. | These are high level actions. |
| Takes more time | Takes less time |
| It works on “how to be tested” | It works on “what to be tested” |

**What is RTM (Requirement Traceability Matrix)? OR (How do you ensure if test case cover all the requirements?)**

Generally, this is like a worksheet document, which contains a table, but there are also many user-defined templates for the traceability matrix. Each requirement in the traceability matrix is connected with its respective test case so that tests can be carried out sequentially according to specific requirements. It is prepared before the test execution to make sure that every requirement is covered in test cases so that nothing is missed in testing.

We map all requirement and corresponding test case to make sure that we have written all the test cases for every condition.

**What is a test plan and what does it include?**

A test plan stores all possible testing activities to ensure a quality product. It gathers data from the product description, requirement, and use case documents.

The test plan document includes the following:

* Testing objectives
* Test scope
* Testing the frame
* Environment
* Reason for testing
* Criteria for entrance and exit
* Deliverables
* Risk factors

**What are the test Environments?**

Test Environments are collection of software and hardware which help us to execute the test case. Once we get the requirement from the customer, the developer starts developing the code. When the coding phase is completed, the application should be installed in the test server. And that application can be accessible with the help of the URL, and the URL could be like this: **HTTP://COMPUTER NAME/PROJECT/COMPANY NAME**

Here, we have four types of servers, which are as follows:

* **Production server**
* **Development server**
* **QA Main server**
* **Staging server**

**What are Different types of SDLC models?**

* **Waterfall Model -**
* **Spiral Model -**
* **V - Model -**
* **Prototype Model -**

**Level of Software Testing**

In order to test any application, we need to go through all the above phases of SDLC. Like SDLC, we have multiple levels of testing, which help us maintain the quality of the software.

 The purpose of Levels of testing is to make software testing systematic and easily identify all possible test cases at a particular level.

There are mainly four **Levels of Testing** in software testing:

1. **Unit Testing**: checks if software components are fulfilling functionalities or not. (Test individual component)
2. **Integration Testing**: checks the data flow from one module to other modules. (Test integrated Component)
3. **System Testing**: evaluates both functional and non-functional needs for the testing. (Test the entire System)
4. **Acceptance Testing**: checks the requirements of a specification or contract are met as per its delivery. (Test the final system.)

**Other types of Testing**:

* Regression Testing
* Buddy Testing
* Alpha Testing
* Beta Testing

**1.Unit Testing:**

**Unit testing** is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not. This is also first level of functional testing, and these are done by the mainly developer or the tester who have programming knowledge.

The first level of testing involves **analysing each unit or an individual component** of the software application.

**2.Integration Testing:**

This is done after the unit testing. When different software module is integrated together, integration testing comes in the picture.

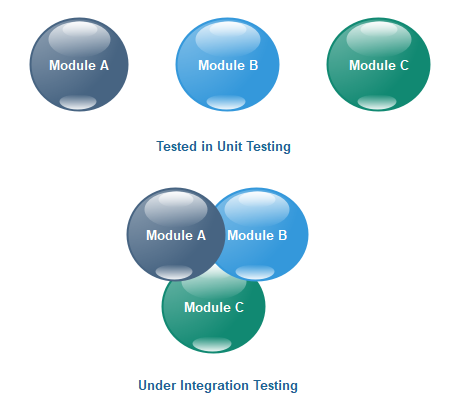
Integration testing make sure each module is communicating with another module as per the requirement after integration.

In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

Let’s understand it more clearly:

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 communication 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**.

**3.System testing:**

This is known as End-to-End testing as in system testing, we test the software as a whole system.

In system testing we will check end to end flow of the software and it falls under black box testing.

When we test the Hardware and software both the requirements for a software that is known as system testing. System Testing includes testing of a fully integrated software system.

A computer system is consisting of group of software to perform various task but only software cannot perform the task; software must be interfaced with compatible hardware. So, system testing validates the full working of an integrated software computer system.

**4.Acceptance Testing:**

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done.**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

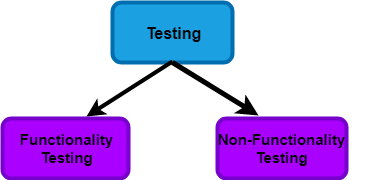
Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

We can say that, UAT is done by the client before accepting the product to check if application is working properly

**Types of Software Testing**

**20. What are the functional and non-functional testing?**

There are two types of testing:



**1. Functional testing** –

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification.

Functional testing comes under black-box testing, because it focuses on application specification rather than actual code

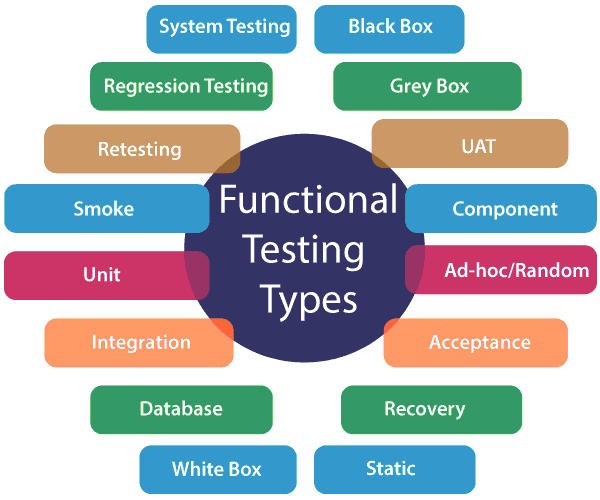
**2.Non-Functional** **Testing**: –

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software.

For example, non-functional testing would be to test how many people can work simultaneously on any software.

**Types of Function Testing: -**

Here are the following types of functional testing.



**1.1 Component Testing**:

In this each component is tested separately without integrating with other part. This is a type of black box testing and also refers as unit testing, programming testing or module testing.

It’s performed by the tester after the unit testing and in that tester will test the application in details of each module/functionality.

**What is build:** In software development, a **build** is the process of converting source code files into standalone software artifact(s) that can be run on a computer.

* 1. **Smoke testing OR build verification testing OR initial level of testing OR Surface level of testing OR Confidence Testing:**

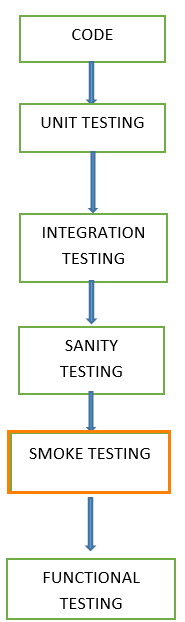
In smoke testing we verify whether the important features are working and there are no stoppers in the build that is under testing. i.e. here we will verify if the build is stable or not.

**Smoke Testing** determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

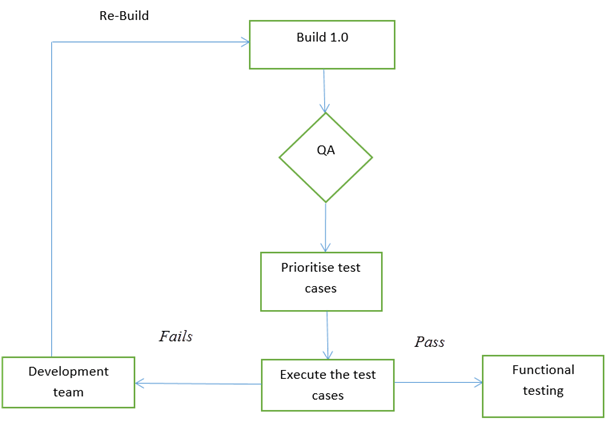
The main purpose of smoke testing is to fail a software with defects so that QA team does not waste time testing broken software application.

**Example**: -New registration button is added in the login window and build is deployed with the new code. We perform smoke testing on a new build.

Check below diagram:



Smoke Testing cycles



* 1. **Sanity Testing:**

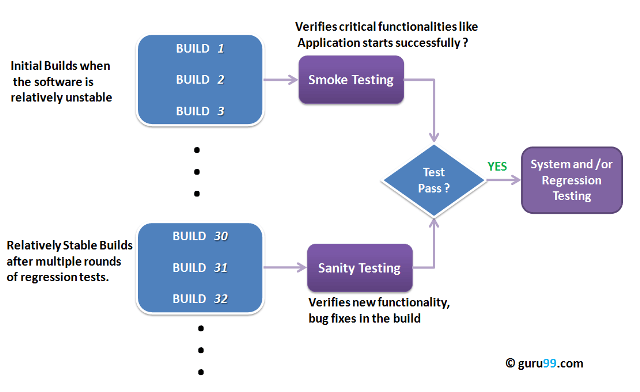
Sanity testing is performed when we are receiving software build (with minor code changes) from the development team. It is a checkpoint to evaluate if testing for the build can proceed or not. In other words, we can say that sanity testing is performed to make sure that all the defects have been solved and no new issues occurs due to minor code changes.

If sanity test fails, the build is rejected to save the time and costs involved in a further testing.

When our build is stable, then we will perform sanity testing to test the newly added functions. Sanity testing is a variant of regression testing.

The purpose of sanity testing is “not” to verify systematically the new functionality but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

Check below diagram that is very useful to under difference between Smoke and Sanity Testing:



**Some man difference between Smoke and Sanity Testing:**

* Smoke Testing has a goal to verify “stability” whereas Sanity Testing has a goal to verify “rationality”.
* Smoke Testing is done by both developers or testers whereas Sanity Testing is done by testers.
* Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
* Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
* Smoke testing is documented or scripted whereas Sanity testing isn’t.
* Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component.
  1. **Regression Testing:**

When there is a change in the original functionality, Regression Testing ensures that the code still works even when the changes are occurring. Regression means Re-test those parts of the application, which are unchanged.

Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs

One should not be confused with regression testing with Re-Testing. Regression testing is done when a change occurs in application to check functionality of previous are unaffected. Where Re-Testing is done when any defect is fixed.

* + 1. **Re-Testing:**

Re-testing is a type of testing which performed to check the test-cases that were unsuccessful in the final execution are successfully pass after the defects repaired.

1. **Difference between Regression and Re-Testing:**

|  |  |
| --- | --- |
| **Regression Testing** | **Re-Testing** |
| Regression Testing is done to confirm whether the code change has not affected the existing features. | Re-testing is performed to ensure that the test cases that are failed in the final execution are passing after the defects fixed. |
| The purpose of regression testing is to ensure that the code changes adversely not affect the existing functionality. | Re-Testing works on defect fixes. |
| Regression testing does not include defect verification | Defect verification is the part of the Retesting. |
| Based on the project type and availability of resources, regression testing can be parallel to Retesting. | The priority of Retesting is higher than Regression Testing, so it is done before the Regression Testing. |
| Regression testing is a generic Testing. | Re-Test is a planned Testing. |
| We can do automation for regression testing; manual testing could be expensive and time-consuming. | We cannot automate the test-cases for Retesting. |
| Regression testing is for passed Test-cases. | Re-testing is for failed test-cases. |
| Regression testing checks for unexpected side effect | . Re-testing make sure that the original fault is corrected. |
| Regression testing is when there is a modification or changes become mandatory in an existing project. | Retesting executes defects with the same data and the same environment with different input with a new build. |
| Regression testing can obtain test cases from the functional specification, user tutorials and manuals, and defects reports in regards to the corrected problem. | Re-testing cannot do before start testing. |

* 1. **Integration Testing:**

This is done after the unit testing. When different software module is integrated together, integration testing comes in the picture. Integration testing make sure each module is communicating with another module as per the requirement after integration. In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

* 1. **GUI Testing (Graphical User Interface):**

It is used to check the Graphical user interface features for the application or the software.

Usually, the GUI testing is used to assesses a design of elements or features like:

* **Text boxes**
* **Font size**
* **Font color**
* **Buttons**
* **Menus**
* **Links**
* **Layout**
* **Labels**
* **Text Formatting**
* **Lists**
* **Captions**
* **Icons**
* **Content**

The primary goal of the GUI testing is to validate the features of the software, or the application performs as per the given requirement/specifications.

The **Graphical user interface testing** process implemented either manually or automatically and repeatedly executed by the **third-parties organization** instead of the developers or the end-users.

In other words, we can say that GUI testing is an approach in which the application's user interface is tested if the software or an application works as expected relating to the user interface performance

* 1. **User Acceptance Testing:**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

We can say that, UAT is done by the client before accepting the product to check if application is working properly

* 1. **Monkey Testing:**

If we don't have enough time to write and perform the tests, we will implement the **monkey testing.** There are many names given to this testing pattern as Random testing, Monkey Testing, Fuzz Testing, and the technical name is Stochastic Testing.

**Monkey Testing** is a software testing technique in which the tester enters any random inputs into the software application without predefined test cases and checks the behaviour of the software application, whether it crashes or not. The purpose of Monkey testing is to find the bugs and errors in the software application using experimental techniques.

1. In Monkey Testing the tester (sometimes developer too) is considered as the ‘Monkey’
2. If a monkey uses a computer, he will randomly perform any task on the system out of his understanding
3. Just like the tester will apply random test cases on the system under test to find bugs/errors without predefining any test case
4. In some cases, Monkey Testing is dedicated to Unit Testing or GUI Testing too



* 1. **Gorilla Testing:**

Gorilla testing is also known as **Frustrating testing, torture testing, and fault tolerance testing.** It is performed repeatedly as it is a part of manual testing.

**1.10 Exploratory Testing:**

If we want to understand any application first, we will perform unit or component testing, suppose the application is having a login page having many elements, and we will understand each part and doing the component testing, but actually, we are doing the exploratory testing because we are exploring the application.

Suppose we have many modules in the application, and we are trying to do some integration scenarios.

Indirectly we are just doing exploratory testing while performing the integration testing.

And, even if we are performing system testing, indirectly, we are performing exploratory testing because here we are also understanding and exploring the application.

To perform exploratory testing, first, we will start using the application and understand the requirement of the application from the person who has a good product knowledge such as senior test engineer, and developers.

Then we will explore the application and write the necessary document, and this document is sent to the domain expert, and they will go through the document.

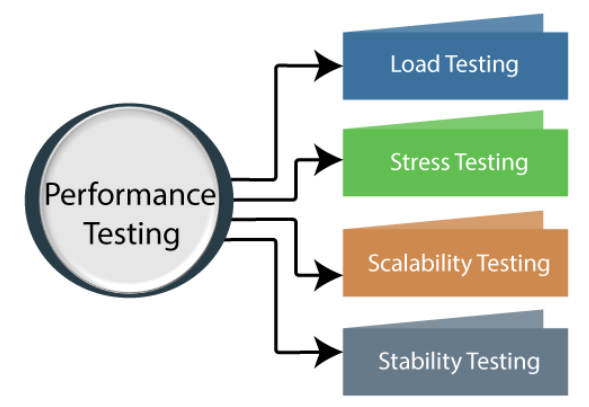
And we can test the application based on our knowledge, and taking the help of the competitive product, which is already launched in the market.

**Types of Non-Functional Testing:**

**2.1 Performance testing:**

In this testing we test the **stability** and **response time** of an application by applying **load**.

* **Response time:** Response time is the time taken by the server to respond to the client's request.
* **Stability:** For the stability factor, we can say that, when N-number of users using the application simultaneously for a particular time.
* **Load:** Here, Load means that when **N-number** of users using the application simultaneously or sending the request to the server at a time.



**2.1.1 Types of Performance Testing:**

**Load testing**:

When you want to determine how many users your system can actually handle, you will perform load testing. You can configure tests to simulate various user scenarios which can focus on different parts of your system (such as a checkout page, for example). You can determine how the load behaves when coming from different geo-locations or how the load might build up, then level out to a sustained level.

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.

**Stress testing:**

The stress testing is testing, which checks the behaviour of an application by applying load greater than the 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.

**Stability testing**

Checking the performance of an application by **applying the load for a particular duration of time** is known as **Stability Testing**.

**Recovery Testing:**

Recovery testing is we test the application to check how well the Software or the application recovers from crashes.

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.

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.

**2.1 Usability Testing:**

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. Typically, the usability testing is executed by real-life users, not by the development team, as sometimes, they fail to identify the more minor defects or bugs related to the user experience.

* 1. **Compatibility Testing:**

Compatibility test is used to ensure a software application is working properly across different browsers, databases, systems (OS), mobile devices, networks and hardware.

Compatibility Testing may have following subtypes:

* **Version testing**- Ensures the software application is [compatible](https://www.techtarget.com/whatis/definition/compatibility) with different versions of the software.
* **Browser testing** - Also known as **cross-browser testing**, this testing ensures the software application performs properly across different browsers -- such as Google Chrome, Firefox, Safari -- as well as across browsers on different devices -- such as laptops, iPhones, Androids and tablets.
* **Hardware testing** - Assesses the performance of the software application with various hardware configurations.
* **Network testing** - Assesses the performance of the software application in different networks, such as 3G, 4G and Wi-Fi.
* **Device testing** - Ensures proper performance of the software application with different devices, such as USB port devices, printers, scanners and Bluetooth.
* **Mobile testing** - Checks if the software application performs with different mobile devices and their various platforms, including iOS and Android OS.
* **OS testing** - Confirms the software application performs appropriately with different operating systems, such as Linux, Mac and Windows.

**Some Other types of Software Testing:**

* 1. **Static Testing:**

**Static Testing** is a type of a Software Testing where we test the application with the help of document without executing the application. Hence this is also called **verification process.**

Whereas in Dynamic Testing checks the code is executed to detect the defects.

Static testing is performed in early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily. The errors that can’t not be found using Dynamic Testing, can be easily found by Static Testing.

* 1. **Dynamic Testing:**

In dynamic testing, we test the application by executing the program. Dynamic testing is done on run time environment by executing the code.

Dynamic testing is the **validation process** where we verify the application by functional and non-functional testing.

**Difference between Verification and Validation**

* **Verification** - Verification is a process to check the client exceptions are meet into the developed software in terms of document and design of the client requirement. It is also known as Static Testing where we ensure that we are developing the right product or not.
* **Validation** – Validation testing is a testing where tester perform functional and non-functional testing. Validation testing is also known as dynamic testing where we are ensuring that “we have developed the right product”.

|  |  |
| --- | --- |
| Verification | Validation |
| We check whether we are developing the right product or not. | We check whether the developed product is right. |
| Verification is also known as static testing. | Validation is also known as **dynamic testing.** |
| Verification includes different methods like Inspections, Reviews, and Walkthroughs. | Validation includes testing like functional testing, system testing, integration, and User acceptance testing. |
| It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements. | It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements. |
| Quality assurance comes under verification testing. | Quality control comes under validation testing. |
| In this type of testing, we can verify that the inputs follow the outputs or not. | In this type of testing, we can validate that the user accepts the product or not. |

**1.3 Alpha Testing:**

**Alpha Testing** is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the **user acceptance testings**. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

**1.4 Beta Testing**

This testing is performed after the Alpha testing and this is a kind of UAT testing. This is the last phase of the testing which is carried out by client or customers.

**Beta Testing** is performed by real users of the application in a real environment. Beta testing is one of the types of **User Acceptance Testing**. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality.

**What is the difference between build and release?**

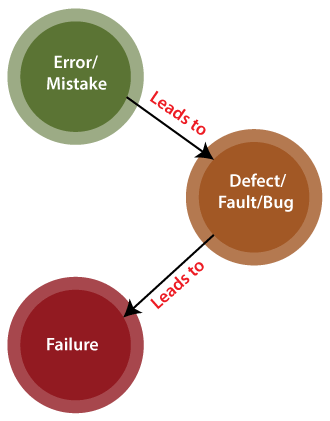
* **Build** – A build is given by the dev teams to test teams We can say that a Build is a software that is still in testing. A **build** is the process of converting source code files into standalone software artifact(s) that can be run on a computer
* **Release** – A release is a formal release of product to its customer/clients. A build when tested and certified by the test team is given to the customers as "**release**". We can say that release is a software that has been tested and released to market.

**What is the difference between error, defect and bug?**

* **Error** - The Problem in code leads to errors. Error happens due to wrong coding; developer use the terms, Error.
* **Bug** - When we have some coding error, it leads a program breakdown, this is known as **a bug. The tester used the terms Bug.**
* **Defect -** Any issue which is found on production or end user site so that is known as defect

**What is static testing? When does it start and what does it cover?**

Static testing is a white-box testing technique that directs developers to verify their code with the help of a checklist to find errors in it. Developers can start the static testing without actually finalizing the application or program. Static testing is more cost-effective than dynamic testing as it more areas than dynamic testing in a shorter time.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Comparison basis** | **Bug** | **Defect** | **Error** | **Fault** | **Failure** |
| **Definition** | It is an informal name specified to the defect. | The **Defect** is the difference between the actual outcomes and expected outputs. | An **Error** is a mistake made in the code; that's why we cannot execute or compile code. | The **Fault** 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. |
| **Raised by** | The **Test Engineers** submit the bug. | The **Testers** identify the defect. And it was also solved by the developer in the development phase or stage. | The **Developers and automation test engineers** raise the error. | **Human mistakes** cause fault. | The failure finds by the manual test engineer through the **development cycle**. |

What are types of defects?

Wrong:

Missing:

Extra:

**Types of Integration Testing:**

* **Incremental integration testing**
* **Non-incremental integration testing**
* **Incremental Testing:**

As we know that in integration testing, we validate if the integrated modules are communication with each other properly. And in Incremental testing we verify the behaviour of the application by integrating all the modules in ascending order one by one as per the customer needs.

* **Non-Incremental Testing:**

When data is created in one module and is combined with all other module, we will perform non incremental testing to verify if all the modules are communicating with each other properly.

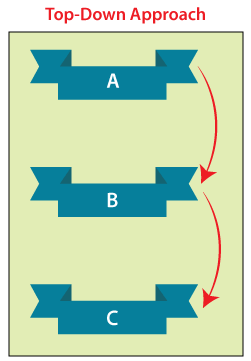
**Incremental integration testing have two types:**

* 1. **Top-Down Integration Testing**
  2. **Bottom-Up integration Testing**

**Top-Down Integration Testing:**

In this testing, all the modules are added incremental or one by one and test the data flow in the similar order. Here high-level modules are tested with low level modules until all the modules have been tested successfully.

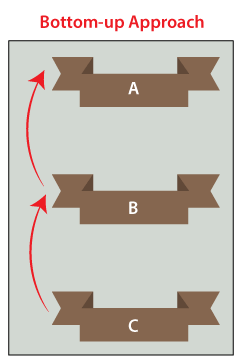
High Level Module Low level module



**Bottom-Up integration testing:**

In this testing, all the modules are added from bottom to the top and test the data flow in the similar manner. Here top-level module is tested at last hence it may cause a defect

High level module Low level module

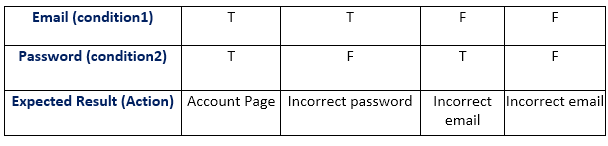


**Technique of Black Box Testing**

1. **Decision table technique:**

This is a systematic approach where various input combinations and their respective system behaviour are captured in a tabular form.

That's why it is also known as a cause-effect table. This technique is related to the correct combination of inputs and determines the result of various combinations of input. To design the test cases by decision table technique, we need to consider conditions as input and actions as output.



**Number of possible conditions = 2^ Number of Values of the second condition**  
**Number of possible conditions =2^2 = 4**

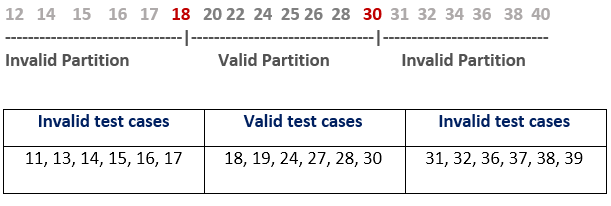
**2. BVA (Boundary value analysis**):

When we test any functionality at the boundary value, that is called BVA testing. For example - Let say we have any tab where we have to entered the mobile no. so in India mobile no. must be in 10 digits always.

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

* 1. For example: if a software accept age only between 18 and 30 then 18 and 30 is boundary value:



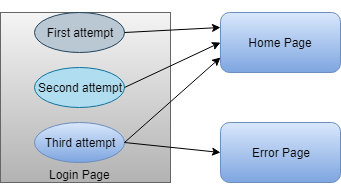
**3. State Transition Technique:**

The general mean of state transition is different form of same situation and state transition method does the same. It is used to capture the behaviour of the software application when different input values are given to the same function.

We all use the ATMs, when we withdraw money from it, it displays account details at last. Now we again do another transaction, then it again displays account details, but the details displayed after the second transaction are different from the first transaction, but both details are displayed by using the same function of the ATM. So, the same function was used here but each time the output was different, this is called state transition

This applies to those types of application that provide the specific number of attempts such as login function which gets locked after specific number of unsuccessful attempts.

Let’s see below example of login functionality that provide maximum three attempts and after three attempts it will redirect to Error page:



|  |
| --- |
| **State transition table** |
| **STATE** | **LOGIN** | **VALIDATION** | **REDIRECTED** |
| S1 | First Attempt | Invalid | S2 |
| S2 | Second Attempt | Invalid | S3 |
| S3 | Third Attempt | Valid  Invalid | S4  S5 |
| S4 | Home Page |  |  |
| S5 | Error Page |  |  |

In the above state transition table, we see that state S1 denotes first login attempt. When the first attempt is invalid, the user will be directed to the second attempt (state S2). If the second attempt is also invalid, then the user will be directed to the third attempt (state S3). Now if the third and last attempt is invalid, then the user will be directed to the error page (state S5).

But if the third attempt is valid, then it will be directed to the homepage (state S4).

**4. All Pair Testing Technique:**

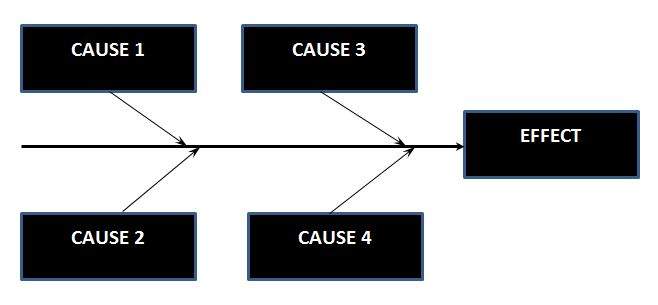
All-pairs testing technique is also known as pairwise testing. It is used to test all the possible **distinct** combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input list box, text box, etc.

Assume we have a piece of software to be tested which has got 10 input fields and 10 possible settings for each input field. Then, there are 10^10 possible inputs to be tested. In this case, exhaustive testing is impossible even if we wish to test all combinations.

**5.Cause and Effect Graph Testing:**

Cause Effect Graph is a black box testing technique that graphically illustrates the relationship between a given outcome and all the factors that influence the outcome.

It is also known as Ishikawa diagram as it was invented by Kaoru Ishikawa or fish bone diagram because of the way it looks.

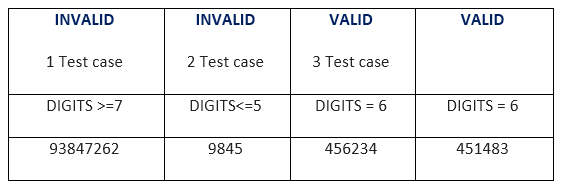


**6**.**Equivalence Partioning Technique:**

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 show the same behaviour. 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.

Test cases should be designed to cover each partition at least once. Each value of every equal partition must show the same behaviour as other.

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



In above example we have two equally valid and invalid partitions

**7. Error Guessing Technique:** -

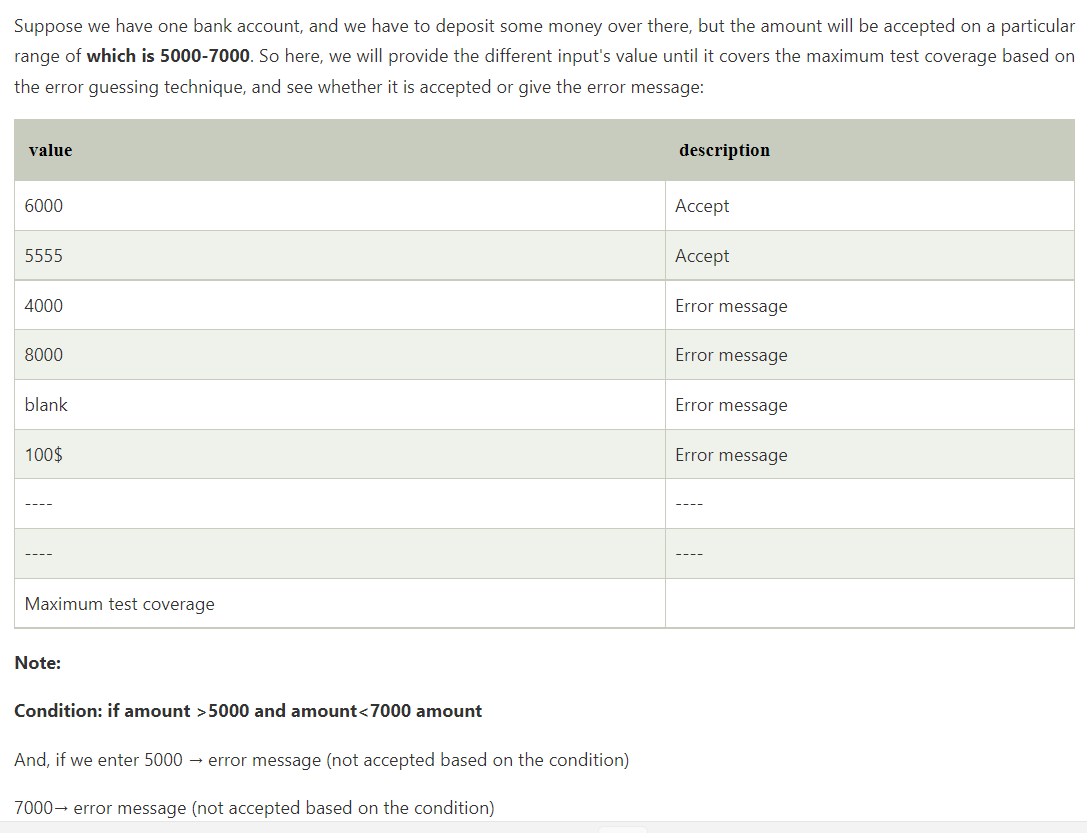
Assumption of the error before starting a testing that is called error guessing

This technique is used to find errors that may not be easily captured by formal black box testing techniques, and that is the reason, it is done after all formal techniques.

The main purpose of this technique is to identify common errors at any level of testing by exercising the following tasks:

* + Enter blank space into the text fields.
  + Null pointer exception.
  + Enter invalid parameters.
  + Divide by zero.
  + Use maximum limit of files to be uploaded.
  + Check buttons without entering values.

**Example of Error guessing:**



**8. Use Case Technique:**

The use case is functional testing of the black box testing 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.

It is a graphic demonstration of business needs, which describe how the end-user will cooperate with the software or the application. The use cases provide us all the possible techniques of how the end-user uses the application as we can see in the below image, that how the **use case** will look like:

A use case is a tool for defining the required user interaction. How the user will use the developed application in reality that is called use case

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.

**Severity and Priority in Testing**

* **Severity:** Impact of any bug on the application is known as Severity. Severity can be **Blocker Critical**, **Major** or **Minor**.
  + **Blocker:** if the severity of a bug is a blocker, which means we cannot proceed to the next module, and unnecessarily test engineer sits ideal.

There are two types of **blocker** bug, which are as follows:

* + - * **A major feature is not working:** Login to HDFC, amount transfer is not working
      * **The major flow is not working:** Login and signup itself not working in HDFC application.
  + **Critical:** if it is critical, that means the main functionality is not working, and the test engineer cannot continue testing.
  + **Major:** if it is major, which means that the supporting components and modules are not working fine, but test engineer can continue the testing.
  + **Minor:** if the severity of a bug is major, which means that all the U.I problems are not working fine, but testing can be processed without interruption.
* **Priority**: To decide which bug should be fixed first or how soon the bug should be fixed. Priority can be **Urgent**, **High**, **Medium** and **Low**.
  + **High:** it is a major impact on the customer application, and it has to be fixed first.
  + **Medium:** In this, the problem should be fixed before the release of the current version in development.
  + **Low:** The flow should be fixed if there is time, but it can be deferred with the next release.

Tester decides the severity and priority for a bug.

**AGILE**

* The Terms agile is linked software development as a project management methodology.
* Agile is an approach which is used to make the continues iteration between testing and development.
* Agile simply means continuous incremental improvement through small and frequent piece of work.

To understand more clearly about Agile read this below short project story:

Agile is an approach which is used to make the continues iteration between testing and development.

*Someone comes up with the idea for a new phone and decides to put a hefty computer chip into this new device. They need an operating system, so a project begins to build the new software. The team lists all the basics they need to make this thing work; along the way, they collect lots of ideas for cool features, and these are formally called* ***issues****. This list of issues is formally labelled a* ***project backlog****.*

*Next, the team has what is called a* ***sprint******planning******meeting****. In the sprint planning, they sort through and prioritize all these issues. The Issues are grouped together into what are known as* ***sprints*** *(a short run) that focus on a specific group of tasks.*

***One sprint contains all the tasks that must be completed*** *just for this new phone to be able to make and receive a call. Another contains all the cool features that will drive potential customers to want to own one of these phones. And still another sprint contains features that will be a little harder to implement, but certainly should be included someday soon. The initial sprint planning produces the first few sprints that, when complete, will allow the release of a working phone with cool new features that make users want to own one.*

*Once the sprint order is set, the team begins working through the first* ***sprint backlog*** *that has all of the things required to get this phone to work. They have a reasonable deadline and they have daily or weekly* ***scrum meetings*** *to make sure they are on task and on time. At or near the deadline, hopefully, they release the Cool Phone Version 0.9. Yea - Beta!*

*Now they start to work on the next sprint backlog that adds all those cool features which will make potential customers want one of these phones. Our heroic team works through the* ***sprint backlog*** *to complete the list of features by the determined deadline, and now they reveal the Cool Phone Version 1.0. Woohoo! We are taking this thing to market!*

*Once available to the public, the customer buys the Cool Phone Version 1.0 because they must own one now! They turn on the Cool Phone and start using it. All is well until they find an issue with how the phone auto dials random people when they put it into their pockets. So the customer logs onto the manufacturer’s website (or uses the Report a Bug function on the phone) and reports a bug about how this phone is misbehaving.*

*The project team adds this new Issue (bug) to the project backlog and they rate it as important. During the next Sprint Planning, the team decides it would be good for the end user (and the manufacturer’s reputation) to make sure this issue (Bug) gets fixed soon, so they put it into the* ***upcoming sprint backlog****.*

*The whole Agile Project Lifecycle repeats as the team goes to work on the newly planned sprint. When the new sprint is completed, the latest version of the Cool Phone Version 1.1 is released. The customer gets a message on their phone saying, “There is an update to your phone. It will add these cool features and it will fix these bugs. Please tap here to update now.” The customer taps to receive the update, the new version is installed, the phone reboots, and the phone no longer experiences problems with pocket dialling. As time goes on, more updates are built and released in the same manner.*

*This, then, is Agile in a nutshell.*

**24. What is framework?**

Scrum and agile are not the same thing because scrum focused on continuous improvement which, is a core foundation of agile. Scrum framework focuses on current getting work done.

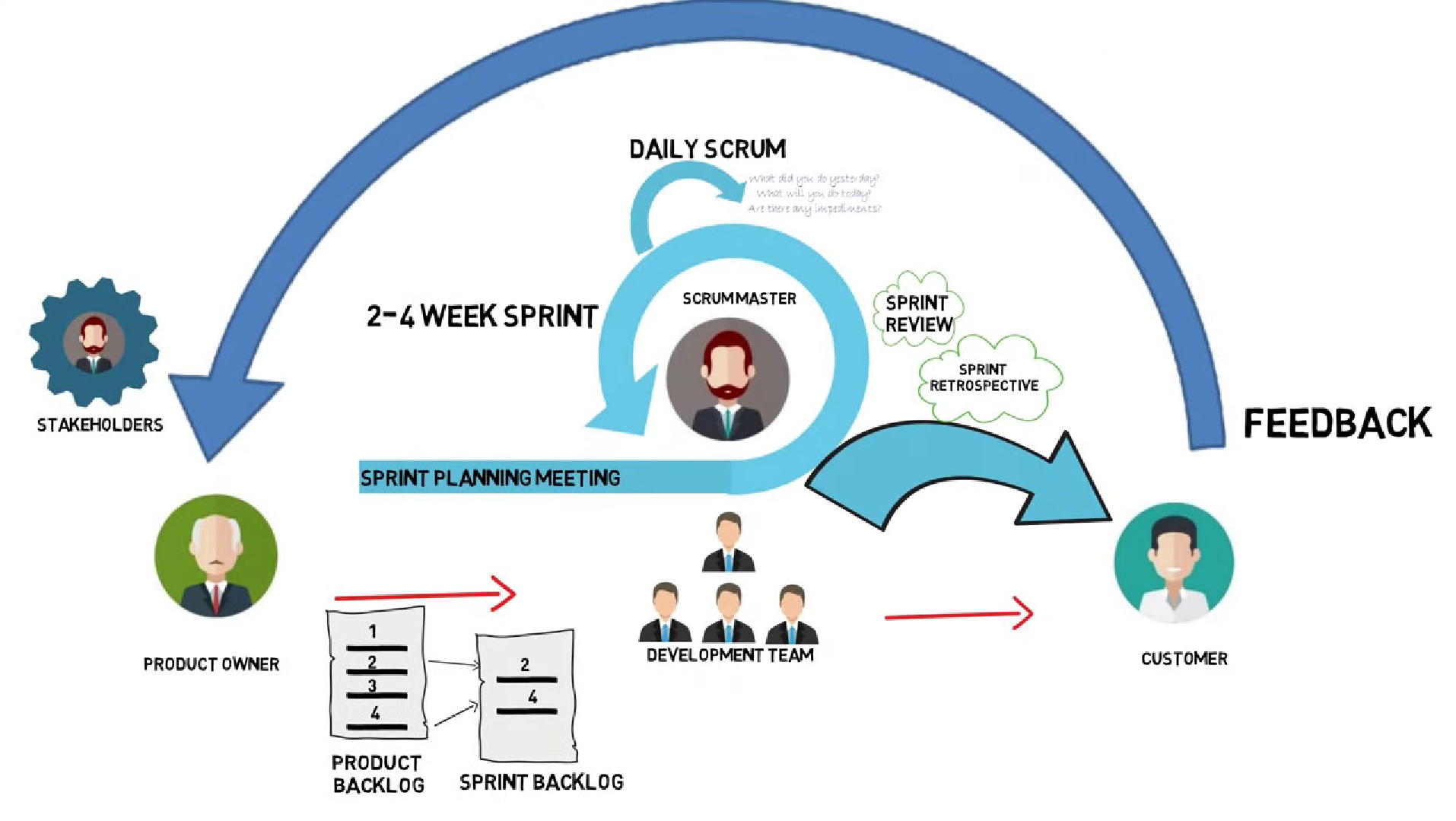
**24. What are the different type of framework in agile?**

**Framework:**

1. Scrum framework
2. Extreme programming (XP)
3. Kanban

We will read only about the Scrum framework.

* **Scrum**
  + Scrum is framework that works on agile methodology. Scrum framework helps agile teams to work together. Using it the team member can deliver and sustain the complex product.
  + Scrum is a work done through the framework and continuously shipping values to customer.



Let’s understand above Diagram:

*In software development the person who decide what kind of product or software is to be built is known as Product Owner. The people who build what is asked by the product owner are known as development team. The product which is built then used by the customer. We know that while following agile value is delivered to customer in small and regular increments. Then customer feedback is taken to understand customer view. The product owner then collects the feedback from the customers. According to all the feedback product owner creates a prioritize list of features which should be added to the software. This prioritize list is known as product backlog. The* development teams add these features to product and this process keep repeating.

Now scrum adds some routine to this process. **The person who helps in facilitating these features is known as the Scrum Master**. A scrum team works on sprints. A sprint is a timebox of one month or less. A sprint planning meeting is *conducted before the sprint which is attended by the scrum master, development team and product owner. All of them select the high priority item from the product backlog in such a way the development team can deliver them in one single sprint. The list of selected items is known as the sprint backlog.*

*During the Sprint every day a daily Scrum Meeting is held, in this meeting each participant answers three question: What did you do yesterday? What will you do today? And what are the obstacles in your way?*

*The outcome of sprint is potential shippable product, it is called potentially shippable product because now it depends on product owner that if he wants to ship the product to the customers or he want some more feature to be added in the product.*

*Sprint review and Sprit Retrospective are the two procedures which occurs post sprint, in sprint review the scrum team shows what is accomplish during the sprint and sprint retrospective scrum team contemplate what went well and what went bad and what could be improved*

**What are agile ceremonies:**

Agile ceremonies are meetings with defined lengths, frequencies, and goals. Their purpose is to help project teams plan, track, and engage stakeholders with their work and help them reflect on how well they’ve worked together. They’re typically a part of the Scrum framework of Agile.

We have total of 4 agile ceremonies:

* Sprint planning meeting.
* Daily stand up meeting.
* Sprint review meeting.
* Sprint retrospective meeting.

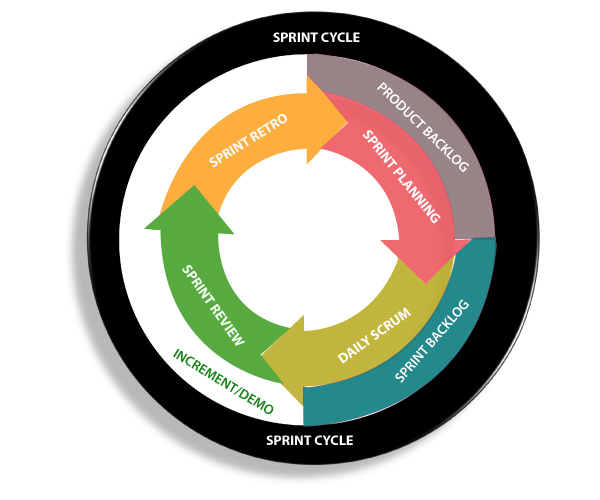
1. **Daily Scrum / Stand Up:**
   1. **The daily stand up is a daily status meeting of agile team member. This may take average of 15 minutes.**
   2. Each member of the team has to answer three important questions

* What he/she did yesterday?
* What he/she will do today?
* The problem he/she is facing . . . He/she blocked due to. . .
  1. **Scrum Master** - Scrum master is a person who leads the daily stand-up call.
     1. **Roles & responsibility of scrum master –**
        1. Every team member should join the Scrum.
        2. Agile process should be followed.
        3. Every team member should get the resolution if any.
        4. Every team member should be known about the task.

1. **Sprint Planning Meeting**
   1. All the idea and work are earlier formally called issue and list of issue is labelled as product backlog. After this a meeting is held that is called Sprint Planning Meeting, where all the team member selects the high prioritize item from the product backlog in such a way that they can be deliver in single sprint. The list of selected items is called sprint backlog. So, we can say that before starting the sprint, a meeting is held known as **sprint planning**, where we will prioritize product backlog known as sprint.
2. **Sprint review and Sprint Retrospective:**

*Sprint review and Sprit Retrospective are the two procedures which occurs post sprint, in sprint review the scrum team shows what is accomplish during the sprint and sprint retrospective scrum team contemplate what went well and what went bad and what could be improved*

**What are Sprints:**

* *With scrum a product is built with a series of repetition called sprints. It breaks down big complex projects into bite-size pieces.*
* *A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies, and getting sprints right will help your agile team ship better software with fewer headaches.*
* *From example, I can also say that in the sprint planning, all the issues (product backlog) are prioritize. The Issues are grouped together into what are known as sprints (a short run usually of two weeks) that focus on a specific group of tasks.*
  + 
  + Sprint should get completed within 2 weeks.
  + **Spill over** - If any task is not completed on the particular sprint so that would complete into the next sprint and that task is called as spill over.

**What are Sprint Backlog and Sprint product?**

* **Sprint Backlog:** Sprint backlog is a list of work items that a team plans to complete during a project sprint

**OR** Sprint backlog is a work which is remaining throughout the sprint.

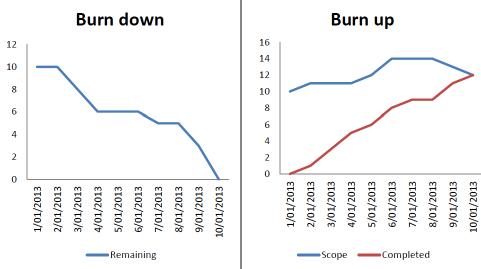
* **Product Backlog:** A product backlog is an ordered list of tasks, features, or items that is needed to improve the project.

**OR** Product backlog is work which is remaining throughout the complete software/application.

**What is Burn Up and Burn Down chart?**

Burn down and burn up charts are two types of charts that are used to track and communicate the progress of the projects

* **Burn UP: A** Burn up shows how much work has been completed and the total amount of work.
* **Burn down:** A burn down charts shows how much work is remaining to be done in project to be completed.



A burn down and burn up chart of the same project. In the burn down chart it appears that the team did not accomplish much in the middle of the project but heroically finished everything at the end. The burn up chart shows the complete picture - that the scope increased at the beginning of the project, and some scope was removed to finish the project by the deadline, whilst the team made steady progress through the entire duration of the project.

**What is Sprint Velocity?**

By looking at the amount of work your team completed in previous sprint, you should be able to estimate how much work they can do in future sprints. In agile development this estimate is known as **Velocity.** So, rate of sprints is known as Velocity.

**SQL (Structured Query Language)**

* Structured data is the data which is stored in the forms of tables. It is a special tool to handle structured data.
* SQL (Structured Query Language) is used to perform operation on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables view, etc.
* SQL is not a database system, but it is a query language.

**SQL Syntax:**

* When we want to do some operation on database, we will have to write predefined syntax of SQL:
* SQL keywords are NOT case sensitive: select is the same as SELECT
* Some important SQL commands: below syntax command ex are from w3sschool.com
  + SELECT - extracts data from a database:
  + UPDATE - updates data in a database
  + DELETE - deletes data from a database
  + INSERT INTO - inserts new data into a database
  + CREATE DATABASE - creates a new database
  + ALTER DATABASE - modifies a database
  + CREATE TABLE - creates a new table
  + ALTER TABLE - modifies a table
  + DROP TABLE - deletes a table
  + CREATE INDEX - creates an index (search key)
  + DROP INDEX - deletes an index

To practice more command, we can use website: <https://www.w3schools.com/sql/sql_syntax.asp>

Suppose we have a database named customer in structured form i.e. data is stored in the forms of tables:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Address** | **City** | **PostalCode** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |
| 4 | Around the Horn | Thomas Hardy | 120 Hanover Sq. | London | WA1 1DP | UK |
| 5 | Berglunds snabbköp | Christina Berglund | Berguvsvägen 8 | Luleå | S-958 22 | Sweden |

**SQL S*ELECT* Statement**

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

*SELECT \* FROM Customers*; [it will select all the table named select customers]

SELECT CustomerName, City FROM Customers; [it will select CustomerName column from the data]

**DISTINCT statement:**

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

*SELECT Country FROM Customers;* [it will select table country from the data]

*SELECT DISTINCT Country FROM* *Customers*; [it will select all the different(distinct) country from the table]

*SELECT COUNT(DISTINCT Country) FROM Customers*; [it will give the count of all different countries.]

**SQL WHERE clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Syntax: SELECT column1, column2, ...  
 FROM table\_name  
 WHERE condition;

Example: SELECT \* FROM Customers  
WHERE Country='Mexico'; [it will select all the customer from the country Mexico]

SELECT \* FROM Customers  
WHERE CustomerID=1; [ it will select all the data from column CustoemrID where value is 1]

**The SQL AND, OR and NOT Operators**

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

**AND syntax:**

SELECT \* FROM Customers  
WHERE Country='Germany' AND City='Berlin'; [*selects* *all* *fields* from "Customers" where country is "Germany" AND city is "Berlin":]

**OR** **syntax**:

SELECT \* FROM Customers  
WHERE City='Berlin' OR City='München'; [selects all fields from "Customers" where country is "Germany" OR "Spain":]

**NOT Syntax:**

SELECT \* FROM Customers  
WHERE NOT Country='Germany'; [select all the field where country is not Germany]

SELECT \* FROM Customers  
WHERE Country='Germany' AND (City='Berlin' OR City='München');[Select all the field where country is Germany and city must be Berlin or Munchen]

**SQL ORDER BY Keyword:**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

**Example:** SELECT \* FROM Customers  
ORDER BY Country; [selects all customers from the "Customers" table, sorted by the "Country" column]

SELECT \* FROM Customers  
ORDER BY Country DESC; [selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column]

SELECT \* FROM Customers  
ORDER BY Country, CustomerName; [selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName]

**The SQL INSERT INTO Statement**

The INSERT INTO statement is used to insert new records in a table.

Example:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

**The SQL NULL Values**

**Example**

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL; [will select CustomerName, ContactName, Address from table customers where address have no value.]

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NOT NULL; [will select CustomerName, ContactName, Address from table customers where address have value]

**SQL UPDATE Statement**

The UPDATE statement is used to modify the existing records in a table.

**Syntax:**

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Example**:

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1; [updates CustomerID = 1 with a new contact person and a new city]

UPDATE Customers  
SET ContactName='Juan'  
WHERE Country='Mexico'; [will update the ContactName to "Juan" for all records where country is "Mexico"]

**SQL DELETE Statement:**

The DELETE statement is used to delete existing records in a table.

**Example:**

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; [deletes the customer "Alfreds Futterkiste" from the "Customers" table]

DELETE FROM Customers; [deletes all rows in the "Customers" table, without deleting the table]

**SQL** **DROP** **Keyword**

DROP is used to delete a whole database or just a table. The DROP statement destroys the objects like an existing database, table, index, or view.  
A DROP statement in SQL removes a component from a relational database management system (RDBMS).

Syntax:

DROP DATABASE Customers; [to delete customers database]

DROP DATABASE City; [to delete City column]

**What is the difference between Delete drop and truncate?**

* **Delete**:
  + - Delete command is used to delete the particular data. With the help of the “**DELETE**” command, we can either delete all the rows in one go or can delete rows one by one. i.e., we can use it as per the requirement or the condition using the Where clause. We can roll back the if we use delete command.
    - Syntax: if we want to delete all the rows of the table will use: **DELETE from;** to delete the row of the table as per the condition then use the Where clause **DELETE from WHERE ;**
* **Drop:**
  + - Drop is used to delete the complete data and structure both.
    - Syntax: If we want to drop the table: **Drop table;**
    - We can’t roll back drop.
* **Truncate:**
  + - Truncate command is used the delete the complete data but structure would remain same**.**
    - Synta**x; Truncate ;**
    - We cant roll back Truncate**.**

**SQL Commands:**

SQL commands are instructions those are used to communicate with the database.

SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

**Types of SQL Commands:**

**I**t is of five types: DDL, DML, DCL, TCL and DQL.

1. **DDL** (Data Definition Language)
2. **DML** (Data Manipulation language)
3. **DCL** (Data Control Language)
4. **TCL** (Transaction control Language)
5. **DQL** (Data Query Language)



**API (Application programming interface)**

* **API** is a computing interface which enables communication and data exchange between two separate software systems OR
* API is an intermediate that allow applications to communicate with each other.
* When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API

**What is API Testing?**

* This is kind of software testing that validate the APIs.
* In API testing, we use software to send signals to API and check the output with expected result.

**Let’s understand the need of Rest API:**

Suppose you are in Bangalore and want to book ticket for a movie called Doctor Strange through BookMyShow app. When you will open the app, you will be able to check movie timing, ticket price, seat allocation etc and all these information are dynamic. This information is coming from a server, most commonly known as webserver.

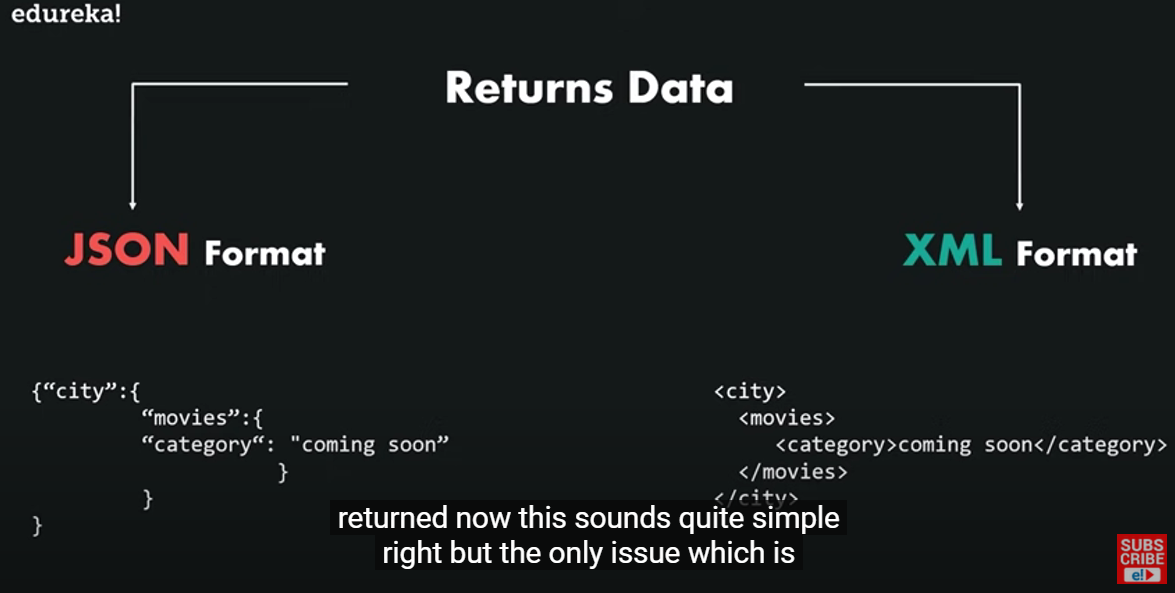
So, a browser is a client and request the required data from the webserver. Webserver, send the desired information in the form of the html webpage.

But sending the data for a specific movie at a specific time at a specific time in html webpage is not suitable response. Data should be returned structured format rather than a complete webpage.

Returned data should be either in JSON format or an XML format.

**JSON** stands for JavaScript Object Notation

XML (Extensible Markup Language) is **a markup language similar to HTML, but without predefined tags to use**



Check above data for movies those are coming soon at a specific city in JSON format and XML format. In case of BookMyShow, a webserver can send the data to the client and this format.

Now look at the code above, JSON is a format of object where object values are returned to the user. And XML format have hierarchical data structured

But we have a problem here, that we will have to use a lot of method to get the desired information and retrieving the data from these methods will be more difficult.

This is where REST API comes into the picture. Rest API creates an object and sent the value of the object to the client (BookMyShow). Now we know what is the need of REST API.

**What are the types of API?**

1. **Rest** –
2. **Soap**–

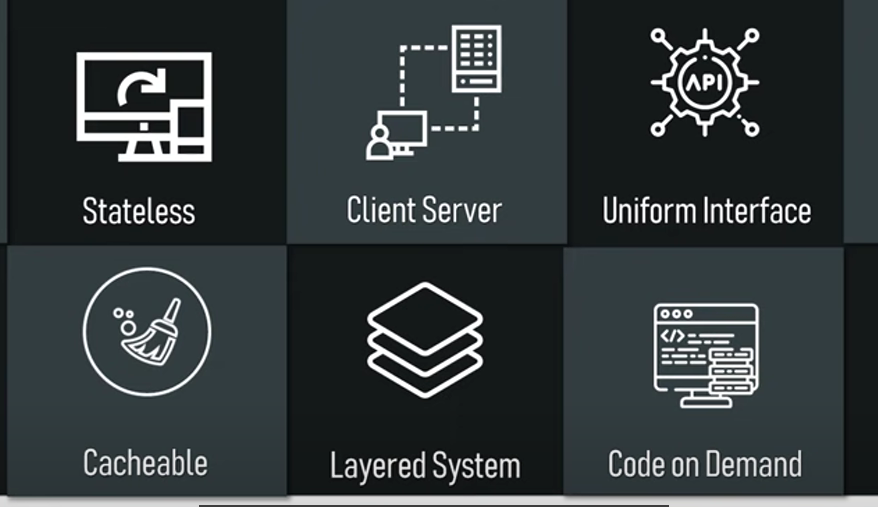
**Rest (Representation State Transfer) API:**

Suppose we want to search for a movie at a certain place and time. Then we can create an object on webserver side, so here an object is created on the webserver and state of that object is being sent to client side from webserver side, that’s why it is known as **RE**presentational **S**tate **T**ransfer.

It is an architectural style as well as an approach for communication purpose that is used in webservice development. In a client server communication REST creates an object of the data requested by the client and send the value of the object to the user.

* *It does not follow any strict standard but follow six constrain.*
* *It uses JSON Language for information exchange.*
* *It is also regarded as “language of the internet”.*
* *Rest is easy to implement and require less bandwidth such as smartphone.*

Rest API has 6 principals:



1. **Soap** –
   1. Stand for Simple Object Access Protocol.
   2. It follows strict standard between client and server.
   3. Soap uses only XML Language for information exchange, which is complicated language.
   4. Soap is difficult to implement and require more bandwidth.

We will learn about the REST services.

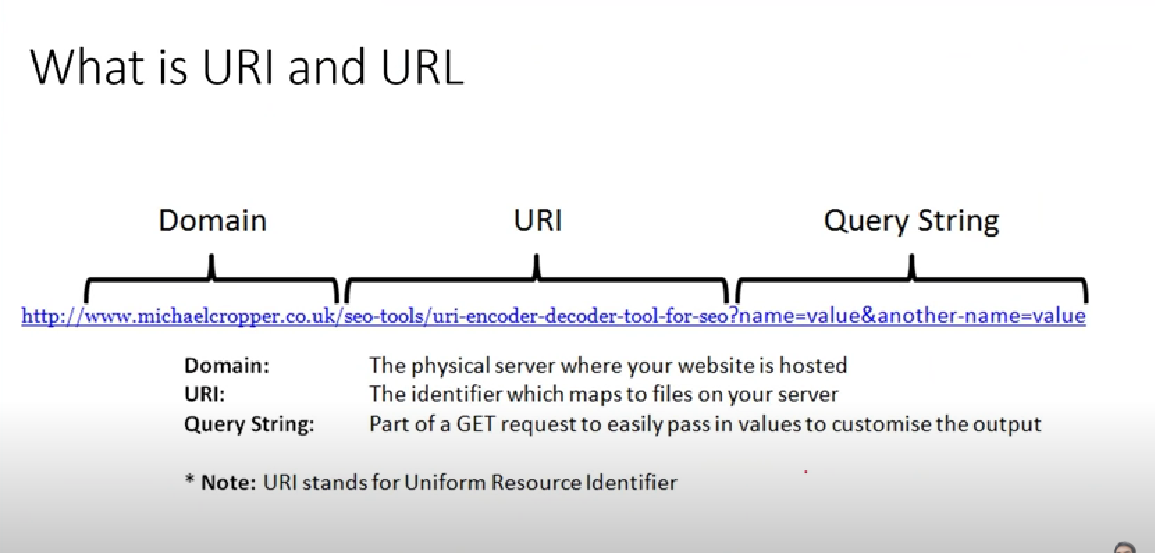
**What are the five methods of API? OR they are also called HTTP Request.**

* **GET** - To fetch the Data ====== Request Body / Request Payload
* **POST** - To Create/Send the Data ====== Request Body / Request Payload
* **PUT** - To update the Data
* **Patch** - To partially update the data
* **Delete** - To delete the data.

**Part of the URL**

Suppose as URL is: <https://reqres.in/api/users?page=2> then

* **https://reqres.in === Domain**
* **/api/users === Path Parameters.**
* **?Page=2 === Query Parameters.** (After ? it is called query parameters.)



We will do manual testing for rest API using Postman tool.

To learn more about Postman tutorial click: [Postman Tutorial](https://www.javatpoint.com/postman)

**HTTPS Status Codes:**

* **1xx Informational** - The request was received but is still processing.
* **2xx Success** - The request was received, accepted and processed successfully.
* **3xx Redirection** - The client needs to take additional action to complete the request.
* **4xx Client Error** - An error occurred on the client side.
* **5xx Server Error** - An error occurred on the server side.

|  |  |  |  |
| --- | --- | --- | --- |
| **1XX: Informational response** | | **2XX: success** | |
| 100 | Continue | **200** | **Ok** |
| 101 | Switching Protocol | **201** | **Created (**Request is completed, new resource is created**)** |
| 102 | Processing | **202** | **Accepted (**The request is accepted for processing, but the processing is not complete) |
| 103 | Early hints | **203** | **Non-Authorities information** (The information in the entity header is from a local or third-party copy, not from the original server) |
|  |  | **204** | **No content (**A status code and a header are given in the response, but there is no entity-body in the reply.  **)** |
|  |  | **205** | **Reset content (**Browser will clear the form data) |
|  |  | **206** | **Partial content** |
|  |  | **207** | **Multi status** |
|  |  | **208** | **Already reported** |
|  |  | **226** | **I am used** |
|  |  |  |  |
| **3XX: Redirection Message** | | **4XX: Client Error** | |
| **300** | **Multiple choice (**A link list. The user can select a link and go to that location. Maximum five addresses  **)** | **400** | **Bad Request (**The Request cannot be fulfilled because of some bad syntax.**)** |
| **301** | **Moved Permanently** (Requested page permanently moved to new URL) | **401** | **Unauthorized (**The request has not been applied because it lacks valid authentication credentials for the target resource) |
| **302** | **Found** | **402** | **Payment required** |
| **304** | **Not modified** | **403** | **Forbidden (**The request is a valid request, but the server is refusing to respond to it due to some factor  **)** |
| **305** | **Use Proxy** | **404** | **Not found** |
| **306** | **Unused** | **405** | **Method not allowed** |
| **307** | **Temporary Redirect** | **406** | **Not acceptable** |
| **308** | **Permanent Redirect** | **407** | **Proxy Authentication Required** |
|  |  | **408** | **Request Timeout** |
|  |  | **409** | **Conflict** |
|  |  | **410** | **Gone** |
|  |  | **411** | **Length Required** |
|  |  | **412** | **Precondition failed** |
|  |  | **413** | **Payload too large** |
|  |  | **414** | **Url too long** |
|  |  | **415** | **Unsupported media type** |
|  |  | **416** | **Range not satisfiable** |
|  |  | **417** | **Expectation failed** |
|  |  | **418** | **I am teapot** |
|  |  | **421** | **Misdirected request** |
|  |  | **422** | **Unprocessable Entity (webdav)** |
|  |  | **423** | **Locked (webdav)** |
|  |  | **424** | **Failed Dependency (webdav)** |
|  |  | **425** | **Too Early** |
|  |  | **426** | **Upgrade Required** |
|  |  | **428** | **Precondition Required** |
|  |  | **429** | **Too many request** |
|  |  | **431** | **Request header field too large** |
|  |  | **451** | **Unavailable for legal reason** |
|  |  |  |  |
| **5XX: Server Error** | |  |  |
| **500** | **Internal server error** |  |  |
| **501** | **Not implemented** |  |  |
| **502** | **Bad Gateway** |  |  |
| **503** | **Service unavailable** |  |  |
| **504** | **Gateway timeout** |  |  |
| **505** | **HTTP version not supported** |  |  |
| **506** | **Variants also negotiates** |  |  |
| **507** | **Insufficient storage** |  |  |
| **508** | **Loop Detected (webdav)** |  |  |
| **510** | **Not extended** |  |  |
| **511** | **Network authentication required.** |  |  |

**1. What is the Difference between put and patch methods?**

PUT is a method of modifying resource where the client sends data that updates the entire resource. PATCH is a method of modifying resources where the client sends partial data that is to be updated without modifying the entire data. In general, we can say that put is used to update the data and patch is used to partially update the data.

**2. What is the difference between Rest and SOAP API?**

**SOAP** uses only XML for exchanging information in its message format whereas **REST** is not restricted to XML and it’s the choice of implementer

**3. What is the Difference between Post and PUT?**

Post is used to create the data where PUT is used to update the data.

**4. What is the difference between Authentication and authorisation API?**

In simple terms, authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to.

OR

* *Authentication* is knowing the identity of the user. For example, Alice logs in with her username and password, and the server uses the password to authenticate Alice.
* *Authorization* is deciding whether a user is allowed to perform an action. For example, Alice has permission to get a resource but not create a resource.

**5. What is AUTH API?**

**6. What is the difference between 200 and 201?**

The **200**-status code is by far the most common returned. It means, simply, that the request was received and understood and is being processed. A **201**-status code indicates that a request was successful and as a result, a resource has been created

**7. What is the difference between 401 and 403?**

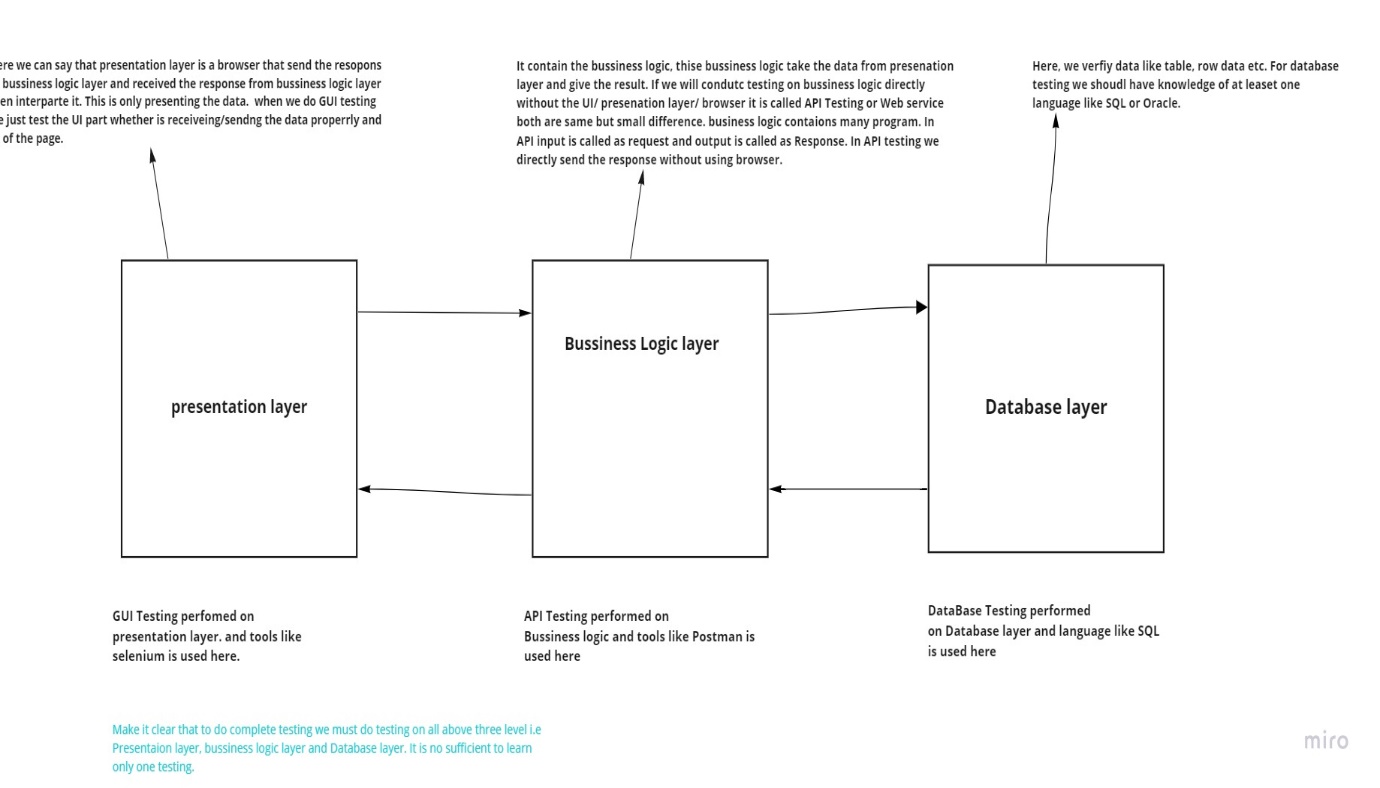
* 401 Unauthorized response status code indicates that the client request has not been completed because it **lacks valid authentication** credentials for the requested resource.
* 403 **Forbidden response** status code indicates that the server understands the request but refuses to authorize it

**8. How to set the Environment Variable?**

**9. What are the types of variables while setting the environment?**

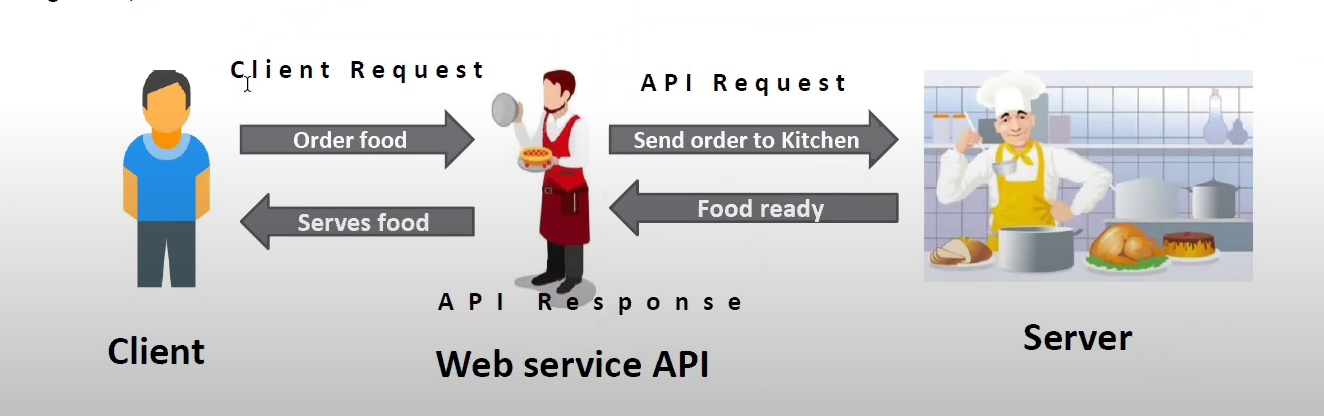
* **Local Variables**: These are temporary variables that will only function inside the environment they were created in. When you change the environment, the variable will stop its access, and an error will occur.
* **Global Variables:** These are independent of environments and function outside the environment.

**All about API in below image:**



**Difference Between APIs and Web Service:**

* Services available over the web are web service.
* All webservices are API. When we put API in the network it called as webservice i.e., when we put the API on the web it is called webservice.
* All API are not webservice because when we test the API it is not necessary to connect it to internet (web) we can test API offline. Once API is moved to web it would be called a webservice.



**Manual Testing FAQs;**

**What is meant by test coverage?**

Test coverage is a quality metric to represent the amount (in percentage) of testing completed for a product. It is relevant for both functional and non-functional testing activities. This metric is used to add missing test cases.

**Is it possible to achieve 100% testing coverage? How would you ensure it?**

It’s considered not possible to perform 100% testing of any product. But you can follow the below steps to come closer.

Set a hard limit on the following factors:

Percentage of test cases passed

Number of bugs found

Set a red flag if:

Test budget is depleted

Deadlines are breached

Set a green flag if:

The entire functionality gets covered in test cases

All critical and major bugs must have a ‘CLOSED’ status

**1. What is SDLC and STLC??**

**2. What is a difference between Project and Product?**

Project is the steps/action to get the product. While product is the final result of action involved in a project.

**3. What is unit testing and who will perform it?**

Unit testing is type of white box testing and developer perform it.

**4. What is Integration testing and who performs it?**

**5. What is difference between QA an QC?**

**6. What is system testing and who performs it?**

**7. What are the types of system testing?**

**8. What is difference between white box and black box testing?**

**9. What is difference between verification and validation?**

**11. What is GUI testing?**

**12. What is input domain testing and what are the technique we will use in this. (Bva and equivalent class partition)**

**13. What is database testing?**

**14. What is difference between load and stress testing.**

**15 What are the test design techniques used while creating test cases?**

**16. Give example of ECP, BVA, Decision table transition diagram.ec**

**17. What is adhoc testing and when this is performed?**

**18. What is bottom-up testing:**

In this testing lowest component are tested first after that higher level of component are tested and this is repeated until the top level

**19. What is the difference between web application and desktop application in the scenario of testing?**

The difference between a web application and desktop application is that a web application is open to the world with potentially many users accessing the application simultaneously at various times, so load testing and stress testing are important. Web applications are also prone to all forms of attacks, mostly DDOS, so security testing is also very important in the case of web applications.

**20. What is the difference between verification and validation?**

**20. What is end to end technique:**

This testing is done end to end, to check what will be a real experience as user. For example, after login to facebook, we will explore some post, like it, comment it, will go to chatbox, will make a call, as so on.

**21. What is difference between test case and test scenario?**

**22. Explain how to report a bug?**

Wont immediately report it to developer, but will test at least three times and will do the prechecks if test data is correct. After being assured, I will raise it on jira tool by specifying the screenshot and all the data related.

**23. What is defect/Bug lifecycles?**

1. **Acceptance Testing:**Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer. Read More on [Acceptance Testing](https://www.guru99.com/user-acceptance-testing.html)
2. **Accessibility Testing:**Type of testing which determines the usability of a product to the people having disabilities (deaf, blind, mentally disabled etc). The evaluation process is conducted by persons having disabilities. Read More on [Accessibility Testing](https://www.guru99.com/accessibility-testing.html)
3. **Active Testing:** Type of testing consisting in introducing test data and analyzing the execution results. It is usually conducted by the testing team.
4. **Agile Testing:**Software testing practice that follows the principles of the agile manifesto, emphasizing testing from the perspective of customers who will utilize the system. It is usually performed by the QA teams. Read More on [Agile Testing](https://www.guru99.com/agile-scrum-extreme-testing.html)
5. **Age Testing:**Type of testing which evaluates a system’s ability to perform in the future. The evaluation process is conducted by testing teams.
6. **Ad-hoc Testing:**Testing performed without planning and documentation – the tester tries to ‘break’ the system by randomly trying the system’s functionality. It is performed by the testing team. Read More on [Ad-hoc Testing](https://www.guru99.com/adhoc-testing.html)
7. **Alpha Testing:**Type of testing a software product or system conducted at the developer’s site. Usually it is performed by the end users. Read More on [Alpha Testing](https://www.guru99.com/alpha-testing.html)
8. **Assertion Testing:**Type of testing consisting in verifying if the conditions confirm the product requirements. It is performed by the testing team.
9. **API Testing:**Testing technique similar to Unit Testing in that it targets the code level. Api Testing differs from Unit Testing in that it is typically a QA task and not a developer task. Read More on [API Testing](https://www.guru99.com/api-testing.html)
10. **All-pairs Testing:**Combinatorial testing method that tests all possible discrete combinations of input parameters. It is performed by the testing teams.
11. **Automated Testing:**Testing technique that uses Automation Testing tools to control the environment set-up, test execution and results reporting. It is performed by a computer and is used inside the testing teams. Read More on [Automated Testing](https://www.guru99.com/automation-testing.html)
12. **Basis Path Testing:**A testing mechanism which derives a logical complexity measure of a procedural design and use this as a guide for defining a basic set of execution paths. It is used by testing teams when defining test cases. Read More on [Basis Path Testing](https://www.guru99.com/basis-path-testing.html)
13. **Backward Compatibility Testing:**Testing method which verifies the behavior of the developed software with older versions of the test environment. It is performed by testing team.
14. **Beta Testing:**Final testing before releasing application for commercial purpose. It is typically done by end-users or others.
15. **Benchmark Testing:**Testing technique that uses representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration. It is performed by testing teams. Read More on [Benchmark Testing](https://www.guru99.com/benchmark-testing.html)
16. **Big Bang Integration Testing:**Testing technique which integrates individual program modules only when everything is ready. It is performed by the testing teams.
17. **Binary Portability Testing:**Technique that tests an executable application for portability across system platforms and environments, usually for conformation to an ABI specification. It is performed by the testing teams.
18. **Boundary Value Testing:**Software testing technique in which tests are designed to include representatives of boundary values. It is performed by the QA testing teams. Read More on [Boundary Value Testing](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html)
19. **Bottom Up Integration Testing:**In bottom-up Integration Testing, module at the lowest level are developed first and other modules which go towards the ‘main’ program are integrated and tested one at a time. It is usually performed by the testing teams.
20. **Branch Testing:**Testing technique in which all branches in the program source code are tested at least once. This is done by the developer.
21. **Breadth Testing:**A test suite that exercises the full functionality of a product but does not test features in detail. It is performed by testing teams.
22. **Black box Testing:**A method of software testing that verifies the functionality of an application without having specific knowledge of the application’s code/internal structure. Tests are based on requirements and functionality. It is performed by QA teams. Read More on [Black box Testing](https://www.guru99.com/black-box-testing.html)
23. **Code-driven Testing:**Testing technique that uses testing frameworks (such as xUnit) that allow the execution of unit tests to determine whether various sections of the code are acting as expected under various circumstances. It is performed by the development teams.
24. **Compatibility Testing:**Testing technique that validates how well a software performs in a particular hardware/software/operating system/network environment. It is performed by the testing teams. Read More on [Compatibility Testing](https://www.guru99.com/compatibility-testing.html)
25. **Comparison Testing:**Testing technique which compares the product strengths and weaknesses with previous versions or other similar products. Can be performed by tester, developers, product managers or product owners. Read More on [Component Testing](https://www.guru99.com/component-testing.html)
26. **Component Testing:**Testing technique similar to unit testing but with a higher level of integration – testing is done in the context of the application instead of just directly testing a specific method. Can be performed by testing or development teams.
27. **Configuration Testing:**Testing technique which determines minimal and optimal configuration of hardware and software, and the effect of adding or modifying resources such as memory, disk drives and CPU. Usually it is performed by the Performance Testing engineers. Read More on [Configuration Testing](https://www.guru99.com/configuration-testing.html)
28. **Condition Coverage Testing:**Type of software testing where each condition is executed by making it true and false, in each of the ways at least once. It is typically made by the Automation Testing teams.
29. **Compliance Testing:**Type of testing which checks whether the system was developed in accordance with standards, procedures and guidelines. It is usually performed by external companies which offer “Certified OGC Compliant” brand.
30. **Concurrency Testing:**Multi-user testing geared towards determining the effects of accessing the same application code, module or database records. It it usually done by performance engineers. Read More on [Concurrency Testing](https://www.guru99.com/concurrency-testing.html)
31. **Conformance Testing:**The process of testing that an implementation conforms to the specification on which it is based. It is usually performed by testing teams. Read More on [Conformance Testing](https://www.guru99.com/conformance-testing.html)
32. **Context Driven Testing:**An Agile Testing technique that advocates continuous and creative evaluation of testing opportunities in light of the potential information revealed and the value of that information to the organization at a specific moment. It is usually performed by Agile testing teams.
33. **Conversion Testing:**Testing of programs or procedures used to convert data from existing systems for use in replacement systems. It is usually performed by the QA teams.
34. **Decision Coverage Testing:**Type of software testing where each condition/decision is executed by setting it on true/false. It is typically made by the automation testing teams.
35. **Destructive Testing:**Type of testing in which the tests are carried out to the specimen’s failure, in order to understand a specimen’s structural performance or material behavior under different loads. It is usually performed by QA teams.  
    Read More on [Destructive Testing](https://www.guru99.com/destructive-testing.html)
36. **Dependency Testing:**Testing type which examines an application’s requirements for pre-existing software, initial states and configuration in order to maintain proper functionality. It is usually performed by testing teams.
37. **Dynamic Testing:**Term used in software engineering to describe the testing of the dynamic behavior of code. It is typically performed by testing teams. Read More on [Dynamic Testing](https://www.guru99.com/dynamic-testing.html)
38. **Domain Testing:**White box testing technique which contains checkings that the program accepts only valid input. It is usually done by software development teams and occasionally by automation testing teams.
39. **Error-Handling Testing:**Software testing type which determines the ability of the system to properly process erroneous transactions. It is usually performed by the testing teams.
40. **End-to-end Testing:**Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate. It is performed by QA teams. Read More on [End-to-end Testing](https://www.guru99.com/end-to-end-testing.html)
41. **Endurance Testing:**Type of testing which checks for memory leaks or other problems that may occur with prolonged execution. It is usually performed by performance engineers. Read More on [Endurance Testing](https://www.guru99.com/endurance-testing.html)
42. **Exploratory Testing:**Black box testing technique performed without planning and documentation. It is usually performed by manual testers. Read More on [Exploratory Testing](https://www.guru99.com/exploratory-testing.html)
43. **Equivalence Partitioning Testing:**Software testing technique that divides the input data of a software unit into partitions of data from which test cases can be derived. it is usually performed by the QA teams. Read More on [Equivalence Partitioning Testing](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html)
44. **Fault injection Testing:**Element of a comprehensive test strategy that enables the tester to concentrate on the manner in which the application under test is able to handle exceptions. It is performed by QA teams.
45. **Formal verification Testing:**The act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics. It is usually performed by QA teams.
46. **Functional Testing:**Type of black box testing that bases its test cases on the specifications of the software component under test. It is performed by testing teams. Read More on [Functional Testing](https://www.guru99.com/functional-testing.html)
47. **Fuzz Testing:**Software testing technique that provides invalid, unexpected, or random data to the inputs of a program – a special area of mutation testing. Fuzz testing is performed by testing teams. Read More on [Fuzz Testing](https://www.guru99.com/fuzz-testing.html)
48. **Gorilla Testing:**Software testing technique which focuses on heavily testing of one particular module. It is performed by quality assurance teams, usually when running full testing.
49. **Gray Box Testing:**A combination of Black Box and White Box testing methodologies: testing a piece of software against its specification but using some knowledge of its internal workings. It can be performed by either development or testing teams.
50. **Glass box Testing:**Similar to white box testing, based on knowledge of the internal logic of an application’s code. It is performed by development teams.
51. **GUI software Testing:**The process of testing a product that uses a graphical user interface, to ensure it meets its written specifications. This is normally done by the testing teams. Read More on [GUI software Testing](https://www.guru99.com/gui-testing.html)
52. **Globalization Testing:**Testing method that checks proper functionality of the product with any of the culture/locale settings using every type of international input possible. It is performed by the testing team. Read More on [Globalization Testing](https://www.guru99.com/globalization-vs-localization-testing.html)
53. **Hybrid Integration Testing:**Testing technique which combines top-down and bottom-up integration techniques in order leverage benefits of these kind of testing. It is usually performed by the testing teams.
54. **Integration Testing:**The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams. Read More on [Integration Testing](https://www.guru99.com/integration-testing.html)
55. **Interface Testing:**Testing conducted to evaluate whether systems or components pass data and control correctly to one another. It is usually performed by both testing and development teams. Read More on [Interface Testing](https://www.guru99.com/interface-testing.html)
56. **Install/uninstall Testing:**Quality assurance work that focuses on what customers will need to do to install and set up the new software successfully. It may involve full, partial or upgrades install/uninstall processes and is typically done by the software testing engineer in conjunction with the configuration manager.
57. **Internationalization Testing:**The process which ensures that product’s functionality is not broken and all the messages are properly externalized when used in different languages and locale. It is usually performed by the testing teams.
58. **Inter-Systems Testing:**Testing technique that focuses on testing the application to ensure that interconnection between application functions correctly. It is usually done by the testing teams.
59. **Keyword-driven Testing:**Also known as table-driven testing or action-word testing, is a software testing methodology for automated testing that separates the test creation process into two distinct stages: a Planning Stage and an Implementation Stage. It can be used by either manual or automation testing teams. Read More on [Keyword-driven Testing](https://www.guru99.com/keyword-driven-testing.html)
60. **Load Testing:**Testing technique that puts demand on a system or device and measures its response. It is usually conducted by the performance engineers. Read More on [Load Testing](https://www.guru99.com/load-testing-tutorial.html)
61. **Localization Testing:**Part of software testing process focused on adapting a globalized application to a particular culture/locale. It is normally done by the testing teams. Read More on [Localization Testing](https://www.guru99.com/localization-testing.html)
62. **Loop Testing:**A white box testing technique that exercises program loops. It is performed by the development teams. Read More on [Loop Testing](https://www.guru99.com/loop-testing.html)
63. **Manual Scripted Testing:**Testing method in which the test cases are designed and reviewed by the team before executing it. It is done by Manual Testing teams.
64. **Manual-Support Testing:**Testing technique that involves testing of all the functions performed by the people while preparing the data and using these data from automated system. it is conducted by testing teams.
65. **Model-Based Testing:**The application of Model based design for designing and executing the necessary artifacts to perform software testing. It is usually performed by testing teams. Read More on [Model-Based Testing](https://www.guru99.com/model-based-testing-tutorial.html)
66. **Mutation Testing:**Method of software testing which involves modifying programs’ source code or byte code in small ways in order to test sections of the code that are seldom or never accessed during normal tests execution. It is normally conducted by testers. Read More on [Mutation Testing](https://www.guru99.com/mutation-testing.html)
67. **Modularity-driven Testing:**Software testing technique which requires the creation of small, independent scripts that represent modules, sections, and functions of the application under test. It is usually performed by the testing team.
68. **Non-functional Testing:**Testing technique which focuses on testing of a software application for its non-functional requirements. Can be conducted by the performance engineers or by manual testing teams. Read More on [Non-functional Testing](https://www.guru99.com/non-functional-testing.html)
69. **Negative Testing:**Also known as “test to fail” – testing method where the tests’ aim is showing that a component or system does not work. It is performed by manual or automation testers. Read More on [Negative Testing](https://www.guru99.com/negative-testing.html)
70. **Operational Testing:**Testing technique conducted to evaluate a system or component in its operational environment. Usually it is performed by testing teams. Read More on [Operational Testing](https://www.guru99.com/operational-testing.html)
71. **Orthogonal array Testing:**Systematic, statistical way of testing which can be applied in user interface testing, system testing, Regression Testing, configuration testing and Performance Testing. It is performed by the testing team. Read More on [Orthogonal array Testing](https://www.guru99.com/orthogonal-array-testing.html)
72. **Pair Testing:**Software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyzes or reviews the testing. This can be done between one Tester and Developer or Business Analyst or between two testers with both participants taking turns at driving the keyboard.
73. **Passive Testing:**Testing technique consisting in monitoring the results of a running system without introducing any special test data. It is performed by the testing team.
74. **Parallel Testing:**Testing technique which has the purpose to ensure that a new application which has replaced its older version has been installed and is running correctly. It is conducted by the testing team.
75. **Ramp Testing:**Type of testing consisting in raising an input signal continuously until the system breaks down. It may be conducted by the testing team or the performance engineer.
76. **Requirements Testing:**Testing technique which validates that the requirements are correct, complete, unambiguous, and logically consistent and allows designing a necessary and sufficient set of test cases from those requirements. It is performed by QA teams.
77. **Storage Testing:**Testing type that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. It is usually performed by the testing team.
78. **Top Down Integration Testing:**Testing technique that involves starting at the top of a system hierarchy at the user interface and using stubs to test from the top down until the entire system has been implemented. It is conducted by the testing teams.
79. **Thread Testing:**A variation of top-down testing technique where the progressive integration of components follows the implementation of subsets of the requirements. It is usually performed by the testing teams.
80. **Upgrade Testing:**Testing technique that verifies if assets created with older versions can be used properly and that user’s learning is not challenged. It is performed by the testing teams.
81. **User Interface Testing:**Type of testing which is performed to check how user-friendly the application is. It is performed by testing teams. Read More on User Interface Testing
82. **Usability Testing:**Testing technique which verifies the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. It is usually performed by end users. Read More on Usability Testing
83. **Volume Testing:**Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner. It is usually conducted by the performance engineer. Read More on Volume Testing
84. **Vulnerability Testing:**Type of testing which regards application security and has the purpose to prevent problems which may affect the application integrity and stability. It can be performed by the internal testing teams or outsourced to specialized companies. Read More on Vulnerability Testing
85. **Workflow Testing:**Scripted end-to-end testing technique which duplicates specific workflows which are expected to be utilized by the end-user. It is usually conducted by testing teams. Read More on Workflow Testing