**Module–2(Manual Testing)**

1. What is Exploratory Testing?

**Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.

Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

1. What is traceability matrix?

Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability.

Traceability can be horizontal through all the test documentation for a given test level or it can be vertical through the layers of development documentation

Types of Traceability Matrix

Forward Traceability – Mapping of Requirements to Test cases

Backward Traceability – Mapping of Test Cases to Requirements

Bi-Directional Traceability - A Good Traceability matrix is the References from test cases to basis documentation and vice versa.

1. What is Boundary value testing?

Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near the limits of valid ranges Boundary value analysis is a method which refines equivalence partitioning. Boundary value analysis generates test cases that highlight errors better than equivalence partitioning. The trick is to concentrate software testing efforts at the extreme ends of the equivalence classes. At those points when input values change from valid to invalid errors are most likely to occur. Boundary Value Analysis (BVA) uses the same analysis of partitions as EP and is usually used in conjunction with EP in test case design

1. What is Equivalence partitioning testing?

Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same. This software testing method divides the input domain of a program into classes of data from which test cases should be designed.

The concept behind this Test Case Design Technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to Identify valid as well as invalid equivalence classes

1. What is Integration testing?

**Integration Testing** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

1. What determines the level of risk?

A factor that could result in future negative consequences; usually expressed as impact and likelihood

Types of Risk

Risks are of two types

• Project Risks

• Product Risk

1. What is Alpha testing?

It is always performed by the developers at the software development site.

Sometimes it is also performed by Independent Testing Team.

Alpha Testing is not open to the market and public

It is conducted for the software application and project.

It is always performed in Virtual Environment.

It is always performed within the organization.

It is the form of Acceptance Testing.

Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers

During this phase, the following will be tested in the application:

• Spelling Mistakes

• Broken Links

• Cloudy Directions

• Alpha Testing is always performed at the time of Acceptance Testing when developers test the product and project to check whether it meets the user requirements or not.

• It is always performed at the developer’s premises in the absence of the users.

• It is considered as the User Acceptance Testing (UAT) which is done at developer’s area.

• Unit testing, integration testing and system testing when combined are known as alpha testing.

1. What is beta testing?

It is always performed by the customers at their own site.

It is not performed by Independent Testing Team.

Beta Testing is always open to the market and public.

It is usually conducted for software product.

It is performed in Real Time Environment.

It is always performed outside the organization.

It is also the form of Acceptance Testing.

Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data.

• It is only a kind of Black Box Testing.

• Beta Testing is always performed at the time when software product and project are marketed.

• It is always performed at the user’s premises in the absence of the development team.

• It is also considered as the User Acceptance Testing (UAT) which is done at customers or users area.

• Beta testing can be considered “pre-release” testing.

1. What is component testing?

It is the first level of testing.

Also known as Unit Testing, Module Testing or Program Testing.

Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”

Component Testing – The testing of individual software components.

Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

1. What is functional system testing?

System Testing - process of testing an integrated system to verify that it meets specified requirements.

There are two types of System Testing which are:

• Functional System Testing

• Non-Functional System Testing

Functional System Testing: -

A requirement that specifies a function that a system or system component must perform

A Requirement may exist as a text document and/or a model

There are two types of techniques

• Requirement Based Functional Testing

• Process Based Testing

1. What is Non-Functional Testing?

System Testing - process of testing an integrated system to verify that it meets specified requirements.

There are two types of System Testing which are:

• Functional System Testing

• Non-Functional System Testing

Non-functional system testing: -

Testing of those requirements that do not relate to functionality

Emphasis on non-functional requirements:

• Performance

• Load

• Data volumes

• Storage

• Recovery

• Usability

• Stress

• Security

1. What is GUI Testing?

Graphical User Interface (GUI) testing is the process of testing the system GUI of the System under Test.

GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

CHECK IN GUI TESTING: -

• Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.

• Check you can execute the intended functionality of the application using the GUI

• Check Error Messages are displayed correctly

• Check for Clear demarcation of different sections on screen

• Check Font used in application is readable

• Check the alignment of the text is proper

• Check the Colour of the font and warning messages is aesthetically pleasing

• Check that the images have good clarity

• Check that the images are properly aligned

• Check the positioning of GUI elements for different screen resolution.

1. What is Adhoc testing?

Adhoc testing is an informal testing type with an aim to break the system.

This testing is primarily performed if the knowledge of testers in the system under test is very high.

Adhoc Testing does not follow any structured way of testing and it is randomly done on any part of application.

Main aim of this testing is to find defects by random checking.

Adhoc testing can be achieved with the testing technique called Error Guessing.

The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.

Types of Adhoc Testing: -

* Buddy Testing - Two buddies mutually work on identifying defects in the same module. Mostly one buddy will be from development team and another person will be from testing team.
* Pair testing - Two testers are assigned modules, share ideas and work on the same machines to find defects. One person can execute the tests and another person can take notes on the findings. Roles of the persons can be a tester and scriber during testing.
* Monkey Testing - Randomly test the product or application without test cases with a goal to break the system.

1. What is load testing?

Load testing: –

It is a performance testing to check system behaviour under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.

Load testing is a kind of performance testing which determines a system performance under real-life load conditions.

This testing helps determine how the application behaves when multiple users access it simultaneously.

1. What is stress Testing?

Stress testing: -

System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.

It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.

Stress Testing is done to make sure that the system would not crash under crunch situations.

Stress testing is also known as endurance testing.

Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity.

Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.

1. What is white box testing and list the types of white box testing?

White Box Testing: -

Testing based on an analysis of the internal structure of the component or system.

White box testing is the detailed investigation of internal logic and structure of the code.

White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.

Types of White Box Testing: -

* Unit testing — tests written as part of the application code, which test that each component is working as expected.
* Mutation testing — a type of unit testing that checks the robustness and consistency of the code by defining tests, making small, random changes to the code and seeing if the tests still pass.
* Integration testing — tests specifically designed to check integration points between internal components in a software system, or integrations with external systems.
* White box penetration testing — an ethical hacker acts as a knowledgeable insider, attempting to attack an application based on intimate knowledge of its code and environment.
* Static code analysis — automatically identifying vulnerabilities or coding errors in static code, using predefined patterns or machine learning analysis.

1. What is black box testing? What are the different black box testing techniques?

Black-box testing: -

Testing, either functional or non-functional, without reference to the internal structure of the component or system.

The testers have no knowledge of how the system or component is structured inside the box.

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.

The tester is oblivious to the system architecture and does not have access to the source code.

Techniques of Black Box Testing: -

* Equivalence partitioning
* Boundary value analysis
* Decision tables
* State transition testing
* Use-case Testing

1. Mention what are the categories of defects?

Software Defect is some kind of error, flaw or some kind of mistake from the development team which prevent the software from the smooth working. It directly affects software quality, software quality is something how smooth and reliable your software is. Smoothness and reliability is how less defects your software have.

Categories of defects:  
Errors of commissions,

Errors of omissions,

Errors of clarity,

Error of speed and capacity.

Error of Commission:  
Commission means instruction or some kind of command given. Now the error in commission means the error in made in command or instruction. For example, suppose I wrote a loop which I was trying to run 10 times but I command it to run more than 10 times by mistake this is the error of commission.

Errors of Omissions:  
As name is already describing error of omission is something which happens accidentally. Omission word means something left out or executed. Practical most common example of this error is suppose we make a function in programming open its bracket but forget to close at the end.

Error of Clarity:  
The most common error in the natural languages. This error happens due to miss understanding between the developer and client. It travels most of the time from the requirements to the software.

Error of Speed or Capacity:  
The name of the error is itself enough i think to tell about it this error. Your software is working fine but not working in the required time this is the error of speed. When it comes to capacity it can be relevant to memory. For example, a small integer is declared where the long integer was required.

1. Mention what big bang testing?

Big bang integration testingis a testing approach where all components or modules are integrated and tested as a single unit. This is done after all modules have been completed and before any system-level testing is performed. This is in contrast to incremental integration testing, in which components are tested one at a time or in small groups. This approach is typically used when there is a tight deadline for delivering the software product, and all development teams are working in parallel on their respective components.   
For example,consider a simple system with three modules A, B, and C. Module A has been tested and found to be working correctly. The same is true for modules B and C. To test the system as a whole, all three modules are integrated and tested together.Features of Big Bang Integration Testing

Features of big bang integration testing:

* **Simulation of the complete system:** Big bang integration testing involves simulating the complete system. This means that all components and modules are integrated and tested at the same time.
* **Testing all components together:** Since all components are integrated and tested at the same time, this means that all components are tested together. This is beneficial as it allows for the testing of interactions between components.
* **No component is left untested:** Since all components are integrated and tested together, this means that no component is left untested. This is beneficial as it ensures that all aspects of the system are tested.
* **Early detection of errors:** Big bang integration testing can detect errors early on in the development process. This is beneficial as it allows for the correction of errors before the system is deployed.
* **Allows for testing of complex interactions:**Big bang integration testing allows for testing of complex interactions between components. This is beneficial as it allows for the identification of errors that may not be detected by other testing methods.
* **Simulate the behaviour of lower-level components:**Big bang integration testing uses stubs and drivers to simulate the behaviour of lower-level components.
* **Top-down approach:**It is also known as a top-down approach because testing starts from the highest level component and moves down the component hierarchy.
* **The basic form of integration testing:** It is the most basic form of integration testing, where all components are integrated and tested together.
* **Risky:**It can be very risky, as all components need to be working correctly for the system to work correctly. This approach is not recommended for large or complex projects.
* **End of the development cycle:** This type of testing is usually done at the end of the development cycle when all the modules are ready to be integrated. It can be used to test the functionality of the system as a whole.
* **Manual:**This type of testing is usually done manually, as it can be difficult to automate all the modules at once.
* **Time-consuming and expensive:** It can be time-consuming and expensive, as all the modules need to be tested together.

1. What is the purpose of exit criteria?

Exit criteria are the defined requirements within software testing that must be met in order to determine that testing has been completed.

* Run out of time
* Run out of budget
* The business tells you it went live last night
* Boss says stop
* All defects have been fixed
* When out exit criteria have been met.

1. When should "Regression Testing" be performed?

Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.

* Change in requirements and code is modified according to the requirement
* New feature is added to the software
* Defect fixing
* Performance issue fix

1. What is 7 key principles? Explain in detail?

7 key principles –

1. Testing shows presence of Defects

2. Exhaustive Testing is Impossible

3. Early Testing

4. Defect Clustering

5. The Pesticide Paradox

6. Testing is Context Dependent

7. Absence of Errors Fallacy

* Testing shows presence of Defects: -

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness. We test to find Faults as we find more defects, the probability of undiscovered defects remaining in a system reduces. Testing cannot prove that there are no defects present.

* Exhaustive Testing is Impossible: -

Testing everything including all combinations of inputs and preconditions is not possible. So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts. For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) tests. This is very unlikely that the project timescales would allow for this number of tests. So, accessing and managing risk is one of the most important activities and reason for testing in any project. We have learned that we cannot test everything (i.e. all combinations of inputs and pre-conditions).

* Early Testing: -

Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives. Testing activities should start as early as possible in the development life cycle These activities should be focused on defined objectives – outlined in the Test Strategy Remember from our Definition of Testing, that Testing doesn’t start once the code has been written.

* Defect Clustering: -

A small number of modules contain most of the defects discovered during pre-release testing, or are responsible for the most operational failures. Defects are not evenly spread in a system They are ‘clustered’ In other words, most defects found during testing are usually confined to a small number of modules Similarly, most operational failures of a system are usually confined to a small number of modules an important consideration in test prioritisation.

* Pesticide Paradox: -

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects. Testing identifies bugs, and programmers respond to fix them as bugs are eliminated by the programmers, the software improves as software improves the effectiveness of previous tests erodes.

* Testing is Context Dependent: -

Testing is basically context dependent. Testing is done differently in different contexts

Different kinds of sites are tested differently.

For example: -

Safety – critical software is tested differently from an e-commerce site. Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software 1 to 3 failures per KLOC typical for industrial software 0.01 failures per KLOC for NASA Shuttle code! Also different industries impose different testing standards

* Absence of Errors Fallacy

If the system built is unusable and does not fulfil the user’s needs and expectations, then finding and fixing defects does not help. If we build a system and, in doing so, find and fix defects .... It doesn’t make it a good system Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations.

1. Difference between QA v/s QC v/s Tester

|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Quality Assurance | Quality Control | Testing |
| 1 | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2 | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| 3 | Process oriented activities. | Product oriented activities. | Product oriented activities. |
| 4 | Preventive activities. | It is a corrective process. | It is a preventive process. |
| 5 | It is a subset of Software Test Life Cycle (STLC) | QC can be considered as the subset of Quality Assurance. | Testing is the subset of Quality Control. |

1. Difference between Smoke and Sanity?

* 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. Difference between verification and Validation?

| Verification | Validation |
| --- | --- |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does not include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |
| It consists of checking of documents/files and is performed by human. | It consists of execution of program and is performed by computer. |
| Verification refers to the set of activities that ensure software correctly implements the specific function. | Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements. |
| After a valid and complete specification the verification starts. | Validation begins as soon as project starts. |
| Verification is for prevention of errors. | Validation is for detection of errors. |
| Verification is also termed as white box testing or static testing as work product goes through reviews. | Validation can be termed as black box testing or dynamic testing as work product is executed. |
| Verification finds about 50 to 60% of the defects. | Validation finds about 20 to 30% of the defects. |
| Verification is based on the opinion of reviewer and may change from person to person. | Validation is based on the fact and is often stable. |
| Verification is about process, standard and guideline. | Validation is about the product. |

1. Explain types of Performance testing?

Performance testing is a type of testing performed to evaluate the different performance attributes of the application like – responsiveness, stability, reliability, etc. For determining these attributes, we have different types of performance testing techniques.

Types of performance testing: -

* [Load testing](https://artoftesting.com/types-of-performance-testing#Load_testing)
* [Stress testing](https://artoftesting.com/types-of-performance-testing#Stress_testing)
* [Endurance testing](https://artoftesting.com/types-of-performance-testing#Endurance_testing)
* [Spike testing](https://artoftesting.com/types-of-performance-testing#Spike_testing)
* [Volume testing](https://artoftesting.com/types-of-performance-testing#Volume_testing)

## Load testing: -

Load testing is a [type of testing](https://artoftesting.com/types-of-testing) which involves evaluating the performance of the system under the expected workload. A typical load test includes determining the response time, throughput, error rate, etc during the course of the load test.  
  
**Example**– For a newly developed application with an anticipated load of around 1000 concurrent users. We will create a load test script and configure it with 1000 virtual users and run it for say 1-hour duration. After the load test completion, we can analyze the test result to determine how the application will behave at the expected peak load.

## Stress testing: -

Stress testing is a type of performance testing where we evaluate the application’s performance at a load much higher than the expected load. Another aspect of the stress testing is to determine the break-point of the application, the point at which the application fails to respond in the correct manner.  
  
**Example** – For an application with an anticipated load of 1000 users we will run the test with 1200 users and check if the application is robust enough to not crash.

## Endurance testing: -

Endurance testing is also known as ‘Soak Testing’. It is done to determine if the system can sustain the continuous expected load for a long duration. Issues like memory leakage are found with endurance testing.  
  
**Example**– For an application like Income tax filing, the application is used continuously for a very long duration by different users. In this type of application, memory management is very critical. For an application like these, we can run the test for 24 hours to 2-days duration and monitor the memory utilization during the whole test execution.

## Spike testing: -

In spike testing, we analyze the behavior of the system on suddenly increasing the number of users. It also involves checking if the application is able to recover after the sudden burst of users.  
  
**Example** – For an e-commerce application running an advertisement campaign, the number of users can increase suddenly in a very short duration. Spike testing is done to analyse these types of scenarios.

## Volume testing: -

The volume testing is performed by feeding the application with a high volume of data. The application can be tested with a large amount of data inserted in the database or by providing a large file to the application for processing. Using volume testing, we can identify the bottleneck in the application with a high volume of data.  
  
**Example** – For a newly developed e-commerce application, we can perform volume testing by inserting millions of rows in the database and then carry out the performance test execution.

27. What is Error, Defect, Bug and failure?

Error: - A mistake in coding is called error.

Defect: - Error found by tester is called defect. The variation between the actual result and expected result is known as defect.

Bug: - defect accepted by development team then is called bug.

Failure: - when a defect reached the end customer it is called failure.

28 Difference between Priority and Severity?

* **Severity: -**  
  Severity is defined as the extent to which a particular defect can create an impact on the software. Severity is a parameter to denote the implication and the impact of the defect on the functionality of the software.
* **Priority: -**  
  Priority is defined as parameter that decides the order in which a defect should be fixed. Defect having the higher priority should be fixed first.

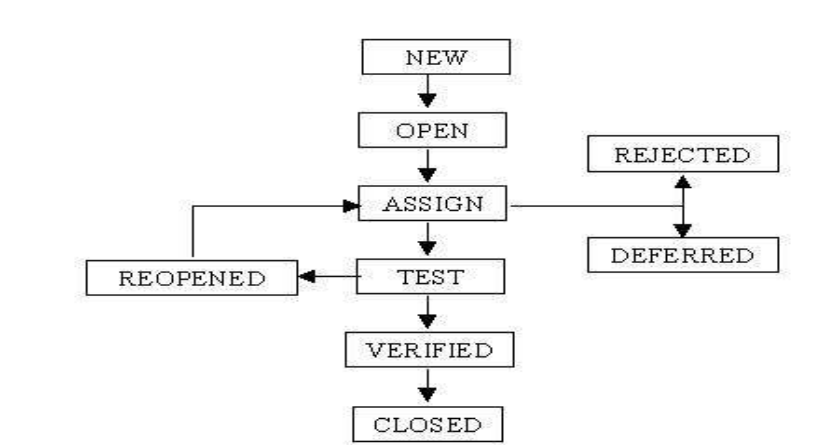
| Severity | Priority |
| --- | --- |
| Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| Severity means how severe defect is affecting the functionality. | Priority means how fast defect has to be fixed. |
| Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| Testing engineer decides the severity level of the defect. | Product manager decides the priorities of defects. |
| Its value is objective. | Its value is subjective. |
| Its value doesn’t change from time to time. | Its value changes from time to time. |
| Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic. | Priority is of 3 types: Low, Medium, and High. |

29. What is Bug Life Cycle?

**Defect Life Cycle** or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life.

The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”



30. Explain the difference between Functional testing and Non-Functional testing?

## Functional Testing -

[Functional testing](https://www.guru99.com/functional-testing.html) is a type of testing which verifies that each **function** of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing, and it is not concerned about the source code of the application.

Every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test. The testing can be done either manually or using automation

## Non-Functional Testing -

Non-functional testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application. It is explicitly designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.

A good example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

#### Difference Between Functional Testing and Non Functional Testing

* Functional testing verifies each function/feature of the software whereas Non Functional testing verifies non-functional aspects like performance, usability, reliability, etc.
* Functional testing can be done manually whereas Non Functional testing is hard to perform manually.
* Functional testing is based on customer’s requirements whereas Non Functional testing is based on customer’s expectations.
* Functional testing has a goal to validate software actions whereas Non Functional testing has a goal to validate the performance of the software.
* A Functional Testing example is to check the login functionality whereas a Non Functional testing example is to check the dashboard should load in 2 seconds.
* Functional describes what the product does whereas Non Functional describes how the product works.
* Functional testing is performed before the non-functional testing.

31. What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?

## SDLC -

Software Development Life Cycle (SDLC) defines all the standard phases which are involved during the software development process. [SDLC life cycle](https://www.guru99.com/software-development-life-cycle-tutorial.html) is a process of developing software through a phased manner in the following order

1. Requirements Gathering
2. Design the software
3. Build the Software
4. Test
5. Deployment
6. Maintenance**.**

Each stage has definite entry and exit criteria along with deliverables.

## STLC -

Software Testing Life Cycle (STLC) is the testing process that is executed in a well-planned manner. In the STLC process, various activities are carried out to improve the quality of the product. However, [STLC phases](https://www.guru99.com/software-testing-life-cycle.html) only deal with testing and detecting errors but not development itself.

Different companies define different phases in STLC. However, generic Software Test Life Cycle has the following stages.

1. Requirement Analysis
2. Test Planning
3. Test Development
4. Test Environment Setup
5. Test Execution & Closure

#### Difference between SDLC and STLC -

* SDLC defines all the standard phases which are involved during the software development process, whereas the STLC process defines various activities to improve the quality of the product.
* SDLC is a Development Life Cycle, whereas STLC is a Testing Life Cycle.
* In SDLC, the development team creates the high and low-level design plans, while In STLC, the test analyst creates the System, Integration Test Plan
* In SDLC, real code is developed, and actual work takes place as per the design documents, whereas in STLC testing team prepares the test environment and executes test cases.
* The SDLC life cycle helps a team complete the software’s successful development, while the STLC phases only cover software testing.

32. What is the difference between test scenarios, test cases, and test script?

Test Scenario -

A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility.

Test Scenario is ‘What to be tested’ Test scenario is nothing but test procedure.

The scenarios are derived from use cases.

Test Scenario represents a series of actions that are associated together.

Scenario is thread of operations

Test Case -

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks.

Test Case is ‘How to be tested’ Test case consist of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition.

Test cases are derived (or written) from test scenario.

Test Case represents a single (low level) action by the user.

Test cases are set of input and output given to the System.

the main components are always available and included in every test case:

Test case ID

Product Module ID

Product version (Optional)

Revision history (Optional)

Purpose/ Test Case Description

Assumptions (Optional)

Pre-Conditions(Optional)

Test Steps

Expected Outcome/Result

Actual Outcome/Result

Post Conditions(Pass/Fail)

Test Script –

The Test Procedures Specification specifies the sequence of actions for a test, i.e. one or more Test Cases It is also known as a Test Script

The Test Script can be manual or automated

Contents of a Test Procedure are:

Test procedure specification identifier

Purpose

Special Requirements

Procedure steps

33. Explain what Test Plan is? What is the information that should be covered?

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

**Test Plan** Process: -

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

34. What is priority?

Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.

Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

Priority Types

Types of Priority of bug/defect can be categorized into three parts:

* Low: The Defect is an irritant but repair can be done once the more serious Defect has been fixed
* Medium: During the normal course of the development activities defect should be resolved. It can wait until a new version is created
* High: The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed

35. What is severity?

**Severity** or Defect Severity in testing is a degree of impact a bug or a[Defect](https://www.guru99.com/defect-management-process.html)has on the software application under test. A higher effect of bug/defect on system functionality will lead to a higher severity level. A[Quality Assurance](https://www.guru99.com/all-about-quality-assurance.html)engineer usually determines the severity level of a bug/defect.

Types of Severity

In Software Testing, Types of Severity of bug/defect can be categorized into the following parts:

* Critical: This defect indicates complete shut-down of the process, nothing can proceed further
* Major: It is a highly severe defect and collapses the system. However, certain parts of the system remain functional
* Medium: It causes some undesirable behavior, but the system is still functional
* Low: It won’t cause any major break-down of the system

36 Bug categories?

A malfunction in the software/system is an error that may cause components or the system to fail to perform its required functions. In other words, if an error is encountered during the test it can cause malfunction. For example, incorrect data description, statements, input data, design, etc.

**1. Lack of Communication:**This is a key factor contributing to the development of software bug fixes. Thus, a lack of clarity in communication can lead to misunderstandings of what the software should or should not do. In many cases, the customer may not fully understand how the product should ultimately work. This is especially true if the software is designed for a completely new product. Such situations often lead to many misinterpretations from both sides.

**2. Repeated Definitions Required:**Constantly changing software requirements creates confusion and pressure in both software development and testing teams. Usually, adding a new feature or deleting an existing feature can be linked to other modules or software components. Observing such problems causes software interruptions.

**3. Policy Framework Does Not Exist:**Also, debugging a software component/software component may appear in a different or similar component. Lack of foresight can cause serious problems and increase the number of distractions. This is one of the biggest problems because of what interruptions occur as engineers are often under pressure related to timelines; constantly changing needs, increasing the number of distractions, etc. Addition, Design and redesign, UI integration, module integration, database management all add to the complexity of the software and the system as a whole.

**4. Performance Errors:**Significant problems with software design and architecture can cause problems for systems. Improved software tends to make mistakes as programmers can also make mistakes. As a test tester, data/announcement reference errors, control flow errors, parameter errors, input/output errors, etc.

**5. Lots of Recycling:**Resetting resources, redoing or discarding a finished work, changes in hardware/software requirements may also affect the software. Assigning a new developer to a project in the middle of nowhere can cause software interruptions. This can happen if proper coding standards are not followed, incorrect coding, inaccurate data transfer, etc. Discarding part of existing code may leave traces on other parts of the software; Ignoring or deleting that code may cause software interruptions. In addition, critical bugs can occur especially with large projects, as it becomes difficult to pinpoint the location of the problem.

37. Advantage of Bugzila.

Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.

Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Center, Testlink etc.

This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data management chain.

Key features of Bugzilla includes –

Advanced search capabilities

E-mail Notifications

Modify/file Bugs by e-mail

Time tracking

Strong security

Customization

Localization

38. Difference between priority and severity?

Severity:  
Severity is defined as the extent to which a particular defect can create an impact on the software. Severity is a parameter to denote the implication and the impact of the defect on the functionality of the software.

Priority:  
Priority is defined as parameter that decides the order in which a defect should be fixed. Defect having the higher priority should be fixed first.

Difference between Severity and Priority in Testing:

| Severity | Priority |
| --- | --- |
| Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| Severity means how severe defect is affecting the functionality. | Priority means how fast defect has to be fixed. |
| Severity is related to the quality standard. | Priority is related to scheduling to resolve the problem. |
| Testing engineer decides the severity level of the defect. | Product manager decides the priorities of defects. |
| Its value is objective. | Its value is subjective. |
| Its value doesn’t change from time to time. | Its value changes from time to time. |
| Severity is of 5 types: Critical, Major, Moderate, Minor, and Cosmetic. | Priority is of 3 types: Low, Medium, and High. |

39. What are the different Methodologies in Agile Development Model?

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Agile is a philosophy, i.e., a set of values and principles to make a decision for developing software. Agile is based on the iterative-incremental model. In an incremental model, we create the system in increments, where each increment is developed and tested individually.

Agile values: -

* Individuals and interactions, Over processes and tools Suppose the team finds any issue in software then they search for another process or tool to resolve the issue. But, in Agile, it is preferable to interact with client, manager or team regarding issue and make sure that the issue gets resolved.
* Working software, Over comprehensive documentation Documentation is needed, but working software is much needed. Agile is not saying that documentation is not needed, but working software is much needed. For example, you have 20-page documents, but you do not have a single prototype of the software. In such a case, the client will not be happy because, in the end, the client needs a document.
* Customer collaboration, Over contract negotiation Contract negotiation is important as they make the budget of software, but customer collaboration is more important than over contract negotiation. For example, if you stuck with the requirements or process, then do not go for a contract which we have negotiated. You need to interact with the customer, gather their requirements.
* Responding to change, over following a plan In the waterfall model, everything is planned, i.e., at what time, each phase will be completed. Sometimes you need to implement the new requirements in the middle of the software, so you need to be versatile to make changes in the software.

Agile Principles

⚫ Customer satisfaction through early and continuous software delivery – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.

⚫ Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.

⚫ Frequent delivery of working software – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.

⚫ Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned.

⚫ Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.

⚫ Enable face-to-face interactions – Communication is more successful when development teams are co-located. 396

⚫ Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.

⚫ Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.

⚫ Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.

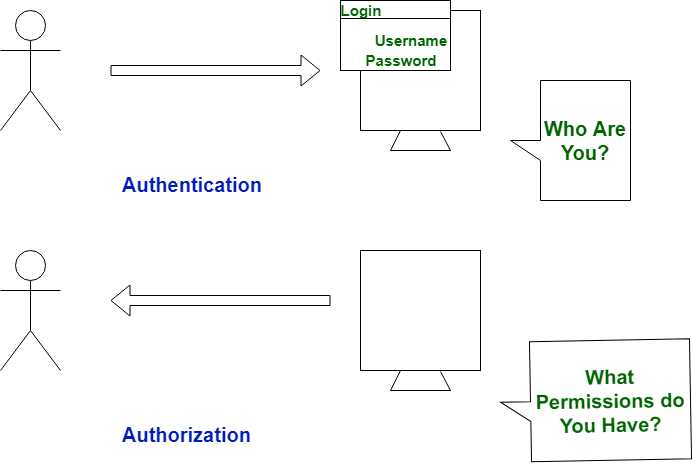
⚫ Simplicity – Develop just enough to get the job done for right now.

⚫ Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.

⚫ Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

40. Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?

Authentication and Authorization area units are utilized in respect of knowledge security that permits the safety of an automatic data system. Each area unit terribly crucial topics usually related to the online as key items of its service infrastructure. However, each of the terms area units is completely different with altogether different ideas. whereas indeed, they’re usually employed in an equivalent context with an equivalent tool, they’re utterly distinct from one another. In the authentication process, the identity of users is checked for providing the access to the system. While in the authorization process, a person’s or user’s authorities are checked for accessing the resources. Authentication is done before the authorization process, whereas the authorization process is done after the authentication process.



Let us see the difference between authentication and authorization:

| Authentication | Authorization |
| --- | --- |
| In the [authentication](https://www.geeksforgeeks.org/authentication-in-computer-network/) process, the identity of users are checked for providing the access to the system. | While in [authorization](https://www.geeksforgeeks.org/what-is-aaa-authentication-authorization-and-accounting/) process, a the person’s or user’s authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user’s login details. | While it needs the user’s privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines **What permission does the user have?** |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. |
| Example: Employees in a company are required to authenticate through the network before accessing their company email. | Example: After an employee successfully authenticates, the system determines what information the employees are allowed to access. |

41. Write a Scenario of Pen –

* Verify the type of pen, whether it is a ballpoint pen, ink pen, or gel pen.
* Verify that the user is able to write clearly over different types of papers.
* Check the weight of the pen. It should be as per the specifications. In case not mentioned in the specifications, the weight should not be too heavy to impact its smooth operation.
* Verify if the pen is with a cap or without a cap.
* Verify the color of the ink on the pen.
* Check the odor of the pen’s ink on writing over a surface.
* Verify the surfaces over which the pen is able to write smoothly apart from paper e.g. cardboard, rubber surface, etc.
* Verify that the text written by the pen should have consistent ink flow without leaving any blob.
* Check that the pen’s ink should not leak in case it is tilted upside down.
* Verify if the pen’s ink should not leak at higher altitudes.
* Verify if the text written by the pen is erasable or not.
* Check the functioning of the pen by applying normal pressure during writing.
* Verify the strength of the pen’s outer body. It should not be easily breakable.
* Verify that text written by pen should not get faded before a certain time as mentioned in the specification.
* Check if the text written by the pen is waterproof or not.
* Verify that the user is able to write normally by tilting the pen at a certain angle instead of keeping it straight while writing.
* Check the grip of the pen, and whether it provides adequate friction for the user to comfortably grip the pen.
* Verify if the pen can support multiple refills or not.
* In the case of an ink pen, verify that the user is able to refill the pen with all the supported ink types.
* For ink pens, verify that the mechanism to refill the pen is easy to operate.
* In the case of a ballpoint pen, verify the size of the tip.
* In the case of a ball and gel pen, verify that the user can change the refill of the pen easily.

44. Write a Scenario of Door –

* Verify if the door is single door or bi-folded door
* Check if the door opens inwards or outwards
* Verify that the dimension of the doors are as per the specifications
* Verify that the material used in the door body and its parts is as per the specifications
* Verify that color of the door is as specified
* Verify if the door is sliding door or rotating door
* Check the position, quality and strength of hinges
* Check the type of locks in the door
* Check the number of locks in the door interior side or exterior side
* Verify if the door is having peek-hole or not
* Verify if the door is having stopper or not
* Verify if the door closes automatically or not – spring mechanism
* Verify if the door makes noise when opened or closed
* Check the door condition when used extensively with water
* Check the door condition in different climatic conditions- temperature, humidity etc
* Check the amount of force- pull or push required to open or close the door

45. Write a Scenario of ATM –

* Verify that all the labels and controls including text boxes, buttons, images, and links are present on the screen.
* Check the informative text written displayed on the screen is clearly visible and legible.
* Verify that the size, color, and UI of the different objects are as per the specifications.
* Verify that the application’s UI is responsive i.e. it should adjust to different screen resolutions of ATM machines.
* Verify the type of ATM machine, if it has a touch screen, both keypad buttons only, or both.
* Verify that on properly inserting a valid card different banking options appear on the screen.
* Check that no option to continue and enter credentials is displayed to the user when the card is inserted incorrectly.
* Verify that the touch of the ATM screen is smooth and operational.
* Verify that the user is presented with the option to choose a language for further operations.
* Check that the user is asked to enter a pin number before displaying any card/bank account detail.
* Verify that there is a limited number of attempts up to which the user is allowed to enter the pin code.
* Verify that if the total number of incorrect pin attempts gets surpassed then the user is not allowed to continue further. And operations like temporary blocking of the card, etc get initiated.
* Check that the pin is displayed in masked form when entered.
* Verify that the user is presented with different account type options like- saving, current, etc.
* Verify that the user is allowed to get account details like available balance.
* Check that the correct amount of money gets withdrawn as entered by the user for cash withdrawal.
* Verify that the user is only allowed to enter the amount in multiple denominations as per the specifications.
* Verify that the user is prompted to enter the amount again in case the amount entered is less than the minimum amount configured.
* Check that the user cannot withdraw more amount than the total available balance and a proper message should be displayed.
* Verify that the user is provided the option to get the transaction details in printed form.
* Verify that the user’s session timeout is maintained.
* Check that the user is not allowed to exceed one transaction limit amount.
* Verify that the user is not allowed to exceed the one-day transaction limit amount.
* Verify that the user is allowed to do only one transaction per pin request.
* Check that in case the ATM machine runs out of money, a proper message is displayed to the user.
* Verify that the applicable fee gets deducted along with the withdrawn amount in case the user exceeds the limit of the number of free transactions in a month.
* Verify that the applicable fee gets deducted along with the withdrawn amount in case the user uses a card of a bank other than that of an ATM.
* Check that the user is not allowed to proceed with the expired ATM card and that a proper error message gets displayed.
* Verify that in case of sudden electricity loss before withdrawing cash, the transaction is marked as null and the amount is not withdrawn from the user’s account.

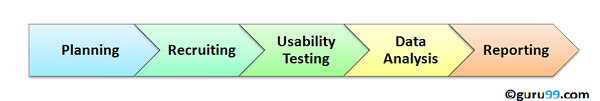
46. When to used Usability Testing?

Usability Testing also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user’s ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.

This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

Usability Testing: Complete Process

Usability testing process consists of the following phases



Phase 1) Planning: During this phase the goals of usability test are determined. Having volunteers sit in front of your application and recording their actions is not a goal. You need to determine critical functionalities and objectives of the system. You need to assign tasks to your testers, which exercise these critical functionalities. During this phase, the usability testing method, number & demographics of usability testers, test report formats are also determined

Phase 2) Recruiting: During this phase, you recruit the desired number of testers as per your usability test plan. Finding testers who match your demographic (age, sex etc.) and professional (education, job etc.) profile can take time.

Phase 3) Usability Testing: During this phase, usability tests are actually executed.

Phase 4) Data Analysis: Data from usability tests is thoroughly analyzed to derive meaningful inferences and give actionable recommendations to improve the overall usability of your product.

Phase 5) Reporting: Findings of the usability test is shared with all concerned stakeholders which can include designer, developer, client, and CEO

Methods of Usability Testing: 2 Techniques

There are two methods available to do usability testing –

1. Laboratory Usability Testing
2. Remote Usability Testing

Laboratory Usability Testing:. This testing is conducted in a separate lab room in presence of the observers. The testers are assigned tasks to execute. The role of the observer is to monitor the behavior of the testers and report the outcome of testing. The observer remains silent during the course of testing. In this testing, both observers and testers are present in a same physical location.

Remote Usability Testing: Under this testing observers and testers are remotely located. Testers access the System Under Test, remotely and perform assigned tasks. Tester’s voice , screen activity , testers facial expressions are recorded by an automated software. Observers analyze this data and report findings of the test

47. What is the procedure for GUI Testing?

Graphical User Interface (GUI) testing is the process of testing the system’s GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.

Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.

Check you can execute the intended functionality of the application using the GUI Check Error Messages are displayed correctly

Check for Clear demarcation of different sections on screen

Check Font used in application is readable Check the alignment of the text is proper

Check the Color of the font and warning messages is aesthetically pleasing

Check that the images have good clarity

Check that the images are properly aligned

Check the positioning of GUI elements for different screen resolution.

Approach of GUI Testing

⚫ MANUAL BASED TESTING Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

⚫ RECORD AND REPLAY GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

⚫ MODEL BASED TESTING A model is a graphical description of system’s behavior. It helps us to understand and predict the system behavior. Models help in a generation of efficient test cases using the system requirements.

48. Write a scenario of Microwave Owen –

* Verify that the dimensions of the oven are as per the specification provided.
* Verify that the oven’s material is optimal for its use as an oven and as per the specification.
* Verify that the oven heats the food at the desired temperature properly.
* Verify that oven heats food at the desired temperature within a specified time duration.
* Verify the ovens functioning with maximum attainable temperature.
* Verify the ovens functioning with minimum attainable temperature.
* Verify that the oven’s plate rotation is speed is optimal and not too high to spill the food kept over it.
* Verify that the oven’s door gets closed properly.
* Verify that the oven’s door opens smoothly.
* Verify the battery requirement of the microwave oven and check that it function’s smoothly at that power.
* Verify that the text written over the oven’s body is clearly readable.
* Verify that the digital display is clearly visible and functions correctly.
* Verify that the temperature regulator is smooth to operate.
* Verify that the temperature regulator works correctly.
* Check the maximum capacity of the oven and test its functioning with that volume of food.
* Check oven’s functionality with different kinds of food – solid, liquid.
* Check the oven’s functionality with different food at different temperatures.
* Verify the oven’s functionality with different kinds of container material.
* Verify that the power cord of the oven is long enough.
* Verify that the usage instruction or user manuals have clear instructions.

49. Write a scenario of Coffee Vending Machine –

* Verify that the dimension of the coffee machine is as per the specification
* Verify that outer body, as well as inner part’s material, is as per the specification
* Verify that the machine’s body color as well brand is correctly visible and as per specification
* Verify the input mechanism for coffee ingredients-milk, water, coffee beans/powder, etc
* Verify that the quantity of hot water, milk, coffee powder per serving is correct
* Verify the power/voltage requirements of the machine
* Verify the effect of suddenly switching off the machine or cutting the power. The machine should stop in that situation and in power resumption, the remaining coffee should not get come out of the nozzle.
* Verify that coffee should not leak when not in operation
* Verify the amount of coffee served in single-serving is as per specification
* Verify that the digital display displays correct information
* Check if the machine can be switched on and off using the power buttons
* Check for the indicator lights when the machine is switched on-off
* Verify that the functioning of all the buttons work properly when pressed
* Verify that each button has an image/text with it, indicating the task it performs
* Verify that complete quantity of coffee should get poured in a single operation, no residual coffee should be present in the nozzle
* Verify the mechanism to clean the system work correctly- foamed
* Verify that the coffee served has the same and correct temperature each time it is served by the machine
* Verify that system should display an error when it runs out of ingredients
* Verify that pressing the coffee button multiple times leads to multiple serving of coffee
* Verify that there is the passage for residual/extra coffee in the machine
* Verify that machine should work correctly in different climatic, moistures and temperature conditions
* Verify that machine should not make too much sound when in operation
* Performance test – Check the amount of time the machine takes to serve a single serving of coffee
* Check the performance of the machine when used continuously until the ingredients run out of the requirements

50. Write a scenario of chair –

* Verify that the chair is stable enough to take an average human load
* Check the material used in making the chair-wood, plastic etc
* Check if the chair’s leg is level to the floor
* Check the usability of the chair as an office chair, normal household chair
* Check if there is back support in the chair
* Check if there is support for hands in the chair
* Verify the paint’s type and color
* Verify if the chair’s material is brittle or not
* Check if cushion is provided with chair or not
* Check the condition when washed with water or effect of water on chair
* Verify that the dimension of chair is as per the specifications
* Verify that the weight of the chair is as per the specifications
* Check the height of the chair’s seat from floor

51. Write a Scenario of Wrist Watch –

* Verify the type of watch – analog or digital.
* In the case of an analog watch, check the correctness time displayed by the second, minute, and hour hand of the watch.
* In the case of a digital watch, check the digital display for hours, minutes, and seconds is correctly displayed.
* Verify the material of the watch and its strap.
* Check if the shape of the dial is as per specification.
* Verify the dimension of the watch is as per the specification.
* Verify the weight of the watch.
* Check if the watch is waterproof or not.
* Verify that the numbers in the dial are clearly visible or not.
* Check if the watch is having a date and day display or not.
* Verify the color of the text displayed in the watch – time, day, date, and other information.
* Verify that clock’s time can be corrected using the key in case of an analog clock and buttons in case of a digital clock.
* Check if the second hand of the watch makes ticking sound or not.
* Verify if the brand of the watch and check if its visible in the dial.
* Check if the clock is having stopwatch, timers, and alarm functionality or not.
* In the case of a digital watch, verify the format of the watch 12 hours or 24 hours.
* Verify if the watch comes with any guarantee or warranty.
* Verify if the dial has glass covering or plastic, check if the material is breakable or not.
* Verify if the dial’s glass/plastic is resistant to minor scratches or not.
* Check the battery requirement of the watch.

52. Write a Scenario of Lift –

* Verify the dimensions of the lift
* Verify the type of door of the lift is as per the specification
* Verify the type of metal used in the lift interior and exterior
* Verify the capacity of the lift in terms of the total weight
* Verify the buttons in the lift to close and open the door and numbers as per the number of floors
* Verify that lift moves to the particular floor as the button of the floor is clicked
* Verify that lift stops when up/down buttons at particular floor are pressed
* Verify if there is an emergency button to contact officials in case of any mishap
* Verify the performance of the floor – the time is taken to go to a floor
* Verify that in case of power failure, lift doesn’t free-fall and get halted in the particular floor
* Verify lifts working in case button to open the door is pressed before reaching the destination floor
* Verify that in case door is about to close and an object is placed between the doors if the doors sense the object and again open or not
* Verify the time duration for which door remain open by default
* Verify if lift interior is having proper air ventilation
* Verify lighting in the lift
* Verify that at no point lifts door should open while in motion
* Verify that in case of power loss, there should be a backup mechanism to safely get into a floor or a backup power supply
* Verify that in case multiple floor number button is clicked, lift should stop at each floor
* Verify that in case of capacity limit is reached users are prompted with warning alert- audio/visual
* Verify that inside lift user is prompted with current floor and direction information the lift is moving towards- audio/visual prompt