# sWhat is exploratory testing?

Exploratory testing is an unscripted approach to software testing, where the tester is free to select any possible methodology to test the software. Exploratory testing is a common practice by software developers that utilize their personal skills and abilities to test the software they developed and/or coded.

Exploratory testing simultaneously tests the functionality and operations of the software while at the same time identifying any functional or technical problems in it. The goal behind exploratory testing is to optimize and improve the software in any way possible.

1. What is traceability matrix?

A traceability matrix is a document that details the technical requirements for a given test scenario and its current state. It helps the testing team understand the level of testing that is done for a given product. The traceability process itself is used to review the test cases that were defined for any requirement.

1. What is boundary value testing?

What is Boundary Value Testing? Boundary value testing is one of the most widely applied black box testing methodologies across the globe. It is defined as a black box test technique that identifies the errors and bugs between the extreme limits or boundaries of a software program or module.

1. What is equivalence partitioning testing?

Equivalence partitioning (EP) is a method for testing software programs. In this technique, the data fed into the software to be tested is divided into partitions of equal sizes. From each partition of data, one test case is needed. The different test cases must test the classes of the software continuously. This helps in the discovery of errors and bugs which may plague the software. Each test case is defined specifically to check a specific type of error. This speeds up the error hunting process as fewer test cases are required.

1. What is integration testing?

Integration testing -- also known as integration and testing is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.

1. What is alpha testing?

Alpha Testing is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. Alpha testing is carried out by the testers who are internal employees of the organization. The main goal is to identify the tasks that a typical user might perform and test them.

To put it as simple as possible, this kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing. The main focus of alpha testing is to simulate real users by using a black box and white box techniques.

1. What is beta testing?

Beta Testing is performed by “real users” of the software application in “real environment” and it can be considered as a form of external User Acceptance Testing. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.

Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

1. What is component testing?

Component testing, also known as program or module testing, is done after unit test this type of testing those test objects can be tested independently as a component without integrating with other components e.g. modules, classes, objects, and programs. This testing is done by the development team.

Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It’s also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

1. What is functional system testing?

Functional testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirement.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

1. What is 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. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing.

Non-functional testing assesses application properties that aren’t critical to functionality but contribute to the end-user experience. Performance and reliability under load aren’t functional components of a software system but can certainly make or break the user experience. Something that fails a non-functional test doesn’t always cause an issue that users would notice, but it can indicate a problem in the system — especially at scale.

1. What is GUI testing?

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons.

1. What is Adhoc testing? Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

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 Software testing technique called Error Guessing. Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

1. What is load testing?

Load Testing is a type of Performance Testing that determines the performance of a system, software product, or software application under real-life based load conditions. Basically, load testing determines the behavior of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions. The load testing is carried out for normal and extreme load conditions.

To maximize the operating capacity of a software application.

To determine whether the latest infrastructure is capable to run the software application or not.

To determine the sustainability of application with respect to extreme user load.

To find out the total count of users that can access the application at the same time.

To determine scalability of the application.

To allow more users to access the application.

1. What is stress testing?

Stress Testing is a software testing technique that determines the robustness of software by testing beyond the limits of normal operation. Stress testing is particularly important for critical software but is used for all types of software. Stress testing emphasizes robustness, availability, and error handling under a heavy load rather than what is correct behavior under normal situations. Stress testing is defined as a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions. It even tests beyond the normal operating point and analyses how the system works under extreme conditions. Stress testing is performed to ensure that the system would not crash under crunch situations. Stress testing is also known as Endurance Testing or Torture Testing.

Stress testing analyzes the behavior of the system after a failure.

Stress testing makes sure that the system recovers after failure.

It checks whether the system works under abnormal conditions.

It ensures to display of appropriate error messages when the system is under stress.

It verifies that unexpected failures do not cause security issues.

It verifies whether the system has saved the data before crashing or not.

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

White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

**white box testing types**

1. Statement Coverage

Statement coverage is one of the pivotal steps involved in the testing process. It offers a whole lot of advantages in terms of execution from time to time.

The process takes place to check whether all the functionalities are working or not. Most of the testers use the step because it is designed to execute all the functions atleast once. As the process starts, we will be able to figure out the possible errors in the web application.

1. Design Coverage

The modern-day software and web applications are not coded in a continuous mode because of various reasons. It is necessary to branch out at some point in time because it helps in segregating effectively.

Branch coverage testing gives a wide room for testers to find quick results. It helps in verifying all the possible branches in terms of lines of code. The step offers better access to find and rectify any kind of abnormal behavior in the application easily.

1. Condition Coverage

This condition coverage reports the true or false outcome of each condition.

Condition coverage measures the condition independently of each other.

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

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems. Black Box Testing Techniques

1. **Equivalence Partitioning**

Testers can divide possible inputs into groups or “partitions”, and test only one example input from each group. For example, if a system requires a user’s birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the “under 18” group and one date in the “over 18” group.

**2. Boundary Value Analysis**

Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

1. **Decision Table Testing**

Many systems provide outputs based on a set of conditions. Testers can then identify “rules” which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.

**4. State Transition Testing**

In some systems, significant responses are generated when the system transitions from one state to another. A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.

If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states. For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.

1. Mention what are the categories of defect?

Design Defects:

The algorithms, login and data elements, module interface, the external software and hardware UI descriptions should be correctly designed. The incompatible or incorrectly designed modules lead to defects in the system.

Command Defects:

An error in the sequences and logic is known as control flow error or command error. The reasons for such defects are missing command, wrong algorithm, incorrect data and code errors.

Boundary Value Defects:

In case the login page is logging in by giving the passport length to 16 characters in the place of 15 characters, then the defect is the boundary value defect.

Error Handling Defects:

The error that is raised while the users interacting with the software need to be handled in the correct flow. The flow should indicate the instruction in the popup message for the mandatory fields to alert the users for incorrect information.

Multithreading Defects:

Executing or running multiple tasks at the time. Complex debugging is possible in the multiple threading process. It may also lead to a system crash/failure due to the condition in deadlock.

Security Defect:

The defects will be different by their nature of the risks. These defects are weaknesses allowing for a potential security attack.

Interface Defects:

The defects in the interactions of the software and the users. Some of the interfaces in the different kinds of forms are complicated interfaces, unclear interfaces and platform based interfaces.

Priority of Defects:

The impact of the bug of an application should be described.

It is the order of priority which the developer will resolve the defects.

The Priority can be changed based on the comparison with other defects.

At the time of UAT, defects are fixed according to the priority.

Priority can be classified

Immediate/Critical :

This generally occurs when the entire functionality of the task is fault and no more testing can do further for the result. Any defects that need immediate attention that affects the flow of the testing comes under this category. Critical severity also fall into the same category.

High :

The defect that does not meet the exit criteria. Due to such defects the testing of the entire application has been stopped until the defects are solved. These defects are resolved once the critical issues are fixed. This kind of defect should be resolved before the release.

Medium :

Defects occur when a particular feature cannot be used the way it should be because of defects in a program, environmental issue. These defects should be fixed once all the critical and serious bugs get fixed. These defects can also be fixed in the next release.

Low :

If few users of the feature encountered a defect such as minor UI issues, spelling mistakes, alignment issues and colour code mismatch are considered as low priority bugs. Sometimes these defects are opened to suggest enhancements in the existing design. This defect does not need any immediate actions as it can be resolved in future.

Severity of Defects:

It is related to the defect fixing urgency.

The testers will set the level of severity.

Once the Severity is fixed it won’t change with time.

It is based on the functionality that the defects affect.

Severity can be classified

Blocker :

When the whole functions feature / functionality missing from the project applications and completely crashes the system will be considered as Blocker and these severity defects are having the highest priority.

Major :

The implemented feature that does not meet the requirements, test cases and behaves differently than the normal flow. It does not cause any system kind of failure but not equal to the blocker but avoiding the unnecessary delay of fixing.

Minor :

The defect will not cause a failure in execution of the applications which is not a major impact. Loss of data falls in the minor severity it also falls in major severity based on the classifications.

Low/Trival :

The valid defect that should be fixed even though there is no impact on the functionality.

Here are some combinations of Priority and Severity:

High Priority and Blocker Severity:

In the login page when the username and password is entered and click submit, instead of logging in the system crashes or else it throws the error message. In the Online site while adding the cart/payment options occurred in the critical times and in the final stage, this issue can impact on business at a great level.

Low Priority and Low Severity:

Spelling mistakes which are considered as minor in the pages. Colour of the button, contents, and hovering options mismatching. Minor alignment issues in the pages that do not affect the flow much.

High Priority and Low Severity:

Displaying the brand logo or the company logo in the company site does not impact the functionalities but it affects the value of the logo that plays the important role in the company website.

Low Priority and High Severity:

A defect found on the social networking site as the beta version of a new feature is released with not many active users using the facilities as today. Though this defect is having a functional defect, as it does not impact the customers directly.

1. What bigbang testing is?

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

Big Bang Testing is an approach of integration testing where integration of all or major components of the system are tested. The Big Bang method is very effective for saving time in the integration testing process. It helps amateur developers to find integration related defects earlier than actual integration testing. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

1. What is the purpose of exit criteria?

Exit criteria in testing are often viewed as a single document commemorating the end of a life cycle phase. It can be defined as “The specific conditions or on-going activities that should be fulfilled before completing the software testing life cycle. STLC specifies which exit criteria is required at each testing phase”. The exit criteria can identify the intermediate deliverables and enable you to track them as independent events.

* The following exit criteria should be considered for completion of a testing phase:

Ensuring all critical Test Cases are passed

Achieving complete Functional Coverage

Identifying and fixing all the high-priority defects

Fixing all the ‘Show Stopper defects’ or ‘Blockers’ and ensuring that none of the identified Critical/Severity 1 defects are in Open Status

Re-testing and closing all the high-priority defects to execute corresponding Regression scenarios successfully

1. When should “regression testing” be performed?

Regression testing is a type of test performed on software that has recently undergone a code or program change to ensure the software and its features are unaffected by the coding changes. Software developers can execute regression testing within a partial or full selection of previously completed test cases, meaning applications that have already undergone modification or testing. They typically execute the previous test cases more than once to make sure that all existing software functionalities and features are working efficiently. Additionally, regression testing can ensure that any new codes added to a software program don't negatively affect the functions and existing features. Regression testing works to introduce new coding seamlessly without disrupting or eliminating old coding within the software.

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

1) Exhaustive testing is not possible

Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application

And the million dollar question is, how do you determine this risk?

To answer this let’s do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle Defect Clustering

2) Defect Clustering

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let’s see this video of the public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

4) Testing shows a presence of defects

testing principle states that – Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software 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.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

5) Absence of Error – fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user’s needs & requirements.

To solve this problem, the next principle of testing states that Early Testing

6) Early Testing

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software’s are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

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

* Quality Assurance

Quality assurance is process oriented. It is all about preventing defects by ensuring the processes used to manage and create deliverables works. Not only does it work, but is consistently followed by the team. Moreover, QA is about engineering processes that assure quality is achieved in an effective and efficient way.

For instance, if a defect is found and fixed, there is no guaranteeing it won’t pop back up. The role of QA is to identify the process that allowed the error to occur and re-engineer the system so that these defects won’t appear for the second time. The QA process verifies that the product will continue to function as the customer expects

Though QC is absolutely necessary, QA is perhaps more important. By the time you reach the QC stage, for instance, fixing bugs becomes an expensive issue. Because of that, focusing efforts on improved QA processes is one of the best investments an organization can make.

Examples of QA include process definition and implementation, training, audits and selection of tools.

* Quality Control

Quality control is product oriented. It is the function of software quality that determines the ending result is what was expected. Whereas QA is proactive, QC is reactive. QC detects bugs by inspecting and testing the product. This involves checking the product against a predetermined set of requirements and validating that the product meets those requirements.

Examples of QC include technical reviews, software testing and code inspections.

* Tester

Testing is a subset of QC. It is the process of executing a system in order to detect bugs in the product so that they get fixed. Testing is an integral part of QC as it helps demonstrate that the product runs the way it is expected and designed for.

To summarize, think of everything as an assembly line. QA can be thought of as the process to ensure the assembly line actually works, while QC is when the products coming off the assembly line are checked to verify they meet the required specifications.

Ultimately, both QA and QC are required for ensuring a successful product. When used together, they can help detect inefficient processes and identify bugs in the product. Moreover, QA and QC can help to develop and deliver a consistently high-quality product to your customers.

1. Difference between smoke test and sanity testing?

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of [Regression Testing](https://www.guru99.com/regression-testing.html) |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

1. Difference between verification and validation?

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| Are we building the system right? | Are we building the right system? |
| **Verification** is the process of evaluating products of a development phase to find out whether they meet the specified requirements. | **Validation** is the process of evaluating software at the end of the development process to determine whether software meets the customer expectations and requirements. |
| The objective of Verification is to make sure that the product being develop is as per the requirements and specifications.design | The objective of Validation is to make sure that the product actually meet up the user’s requirements, and check whether the specifications were correct in the first place. |
| Following activities are involved in **Verification**: Reviews, Meetings and Inspections. | Following activities are involved in **Validation**: Testing like black box testing, white box testing, gray box testing etc. |
| **Verification** is carried out by QA team to check whether implementation software is as per specification document or not. | **Validation** is carried out by testing team. |
| Execution of code is not comes under **Verification**. | Execution of code is comes under **Validation**. |
| **Verification** process explains whether the outputs are according to inputs or not. | **Validation** process describes whether the software is accepted by the user or not. |
| **Verification** is carried out before the Validation. | **Validation** activity is carried out just after the Verification. |
| Following items are evaluated during **Verification**: Plans, Requirement Specifications, Design Specifications, Code, Test Cases etc, | Following item is evaluated during **Validation**: Actual product or Software under test. |
| Cost of errors caught in **Verification** is less than errors found in Validation. | Cost of errors caught in **Validation** is more than errors found in Verification. |
| It is basically manually checking the of documents and files like requirement specifications etc. | It is basically checking of developed program based on the requirement specifications documents & files. |

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.

* Load testing : Load testing is a type 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 testin

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 analyze 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.

1. What is error, defect, bug, and failure?

* Error

An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of the developer, we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one that is generated because of the wrong login, loop or syntax. The error normally arises in software; it leads to a change in the functionality of the program.

* Deffect

It can be simply defined as a variance between expected and actual. The defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with their external behavior or with its internal features. In other words, a Defect is a difference between expected and actual results in the context of testing. It is the deviation of the customer requirement.

* Bug

A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.

* Failure

A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development, Failures are usually observed by testers.

1. Difference between Priority and Severity

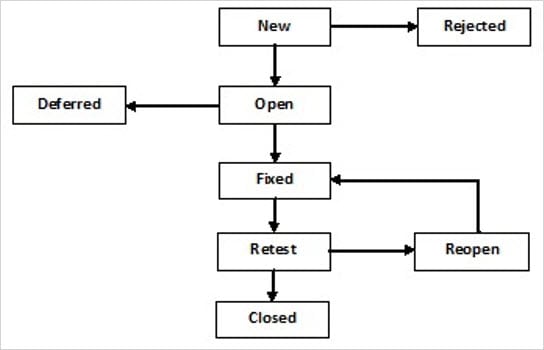
| Severity | Priority |
| --- | --- |
| Severity is a parameter to denote the impact of a particular defect on the software. | Priority is a parameter to decide the order in which defects should be fixed. |
| Severity means how severe defect is affecting the functionality. | Priority means how fast defect 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. |

1. What is bug life cycle?

The Defect Life Cycle, also known as the Bug Life Cycle, is a cycle of defects from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won’t get reproduced again.

Defect Workflow

It is now time to understand the actual workflow of a Defect Life Cycle with the help of a simple diagram as shown below.

Defect Life cycle 

Defect States

1) New: This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a ‘New’ state, and validations & testing are performed on this defect in the later stages of the Defect Life Cycle.

2) Assigned: In this stage, a newly created defect is assigned to the development team to work on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

3) Open: Here, the developer starts the process of analyzing the defect and works on fixing it, if required.

If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely Duplicate, Deferred, Rejected, or Not a Bug-based upon a specific reason. We will discuss these four states in a while.

4) Fixed: When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as “Fixed”.

5) Pending Retest: After fixing the defect, the developer assigns the defect to the tester to retest the defect at their end, and until the tester works on retesting the defect, the state of the defect remains in “Pending Retest”.

6) Retest: At this point, the tester starts the task of retesting the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

7) Reopen: If any issue persists in the defect, then it will be assigned to the developer again for testing and the status of the defect gets changed to ‘Reopen’.

8) Verified: If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to ‘Verified’.

9) Closed: When the defect does not exist any longer, then the tester changes the status of the defect to “Closed”.

A Few More:

* Rejected: If the defect is not considered a genuine defect by the developer then it is marked as “Rejected” by the developer.
* Duplicate: If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect then the status of the defect is changed to ‘Duplicate’ by the developer.
* Deferred: If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as ‘Deferred’.
* Not a Bug: If the defect does not have an impact on the functionality of the application, then the status of the defect gets changed to “Not a Bug”.

1. Explain the different between functional and non-functional testing?

| **Parameters** | **Functional** | **Non-functional testing** |
| --- | --- | --- |
| **Execution** | It is performed before non-functional testing. | It is performed after the functional testing. |
| **Focus area** | It is based on customer’s requirements. | It focusses on customer’s expectation. |
| **Requirement** | It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| **Usage** | Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| **Objective** | Carried out to validate software actions. | It is done to validate the performance of the software. |
| **Requirements** | Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| **Manual testing** | Functional testing is easy to execute by manual testing. | It’s very hard to perform non-functional testing manually. |
| **Functionality** | It describes what the product does. | It describes how the product works. |
| **Example Test Case** | Check login functionality. | The dashboard should load in 2 seconds. |
| **Testing Types** | Examples of Functional Testing Types   * Unit testing * Smoke testing * User Acceptance * Integration Testing * Regression testing | Examples of Non-functional Testing Types   * Performance Testing * Volume Testing * Scalability * Usability Testing * Load Testing * Stress Testing * Portability Testing |

1. To create Test Case of Instagram and facebook

|  |  |  |  |
| --- | --- | --- | --- |
| Instagram  Verify that cursor is focused on the “Username” text box on the page load (login page)  Verify that tab functionality is working properly or not  Verify that Enter/Tab key works as a substitute for the Sign-in button  Verify that the User is able to Login with Valid Credentials  Verify that the User is not able to Login with an invalid Username and invalid Password  Verify that the User is not able to Login with a Valid Username and invalid Password  Verify that the User is not able to log in with an invalid Username and Valid Password  Verify that the User is not able to log in with a blank Username or Password  Verify that the User is not able to Login with inactive credentials  Verify that the reset button clears the data from all the text boxes in the login form  Verify that the login credentials, mainly password stores in a database in an encrypted format  Verify that clicking on the browser back button after successful login should not take the User to log out mode  Verify that validation message is displayed in the case when User leaves Username or Password as blank  Verify that validation message is displayed in case of exceeding the character limit of the Username and Password fields  Verify that validation message is displayed in case of entering special character in the Username and password fields  Verify that the “Keep me logged in” checkbox is unselected by default (depends on business logic, it may be selected or unselected)  Verify that the timeout of the login session (Session Timeout)  Verify that the logout link is redirected to login/home page  Verify that User is redirected to appropriate page after successful login  Verify that the User is redirected to the Forgot password page when clicking on the Forgot Password link  Verify that the User is redirected to the Create an account page when clicking on the Signup / Create an account link  Verify that the User should be able to login with the new password after changing the password  Verify that the user should not be able to login with the old password after changing the password  Verify that spaces should not be allowed before any password characters attempted  Verify whether the user is still logged in after a series of actions such as sign-in, close the browser, and reopen the application.  Verify that the ways to retrieve the password if the user forgets the password    Facebook Test case    Check the login functionality with valid email id and valid password.  Check the login functionality with valid email id and invalid password.  Check the login functionality with invalid email id and valid password.  Check the login functionality with invalid email id and invalid password.  Check the login functionality with blank email id and password.  Check the login functionality with blank email id and valid password.  Check the login functionality with valid email id and blank password.  Check the login functionality with valid phone number and valid password.  Check the login functionality with invalid phone number and valid password.  Check the login functionality with valid phone number and invalid password.  Check the login functionality with blank phone number and valid password.  Check the length of email address and password field.  Check is the error message display when any field is left blank.  Check the Tab key functionality on the Login page.  Check the remember me checkbox functionality  Check is the welcome message is displaying after successfully login into application.  Check the forgotten Password functionality.  Check the password text format is encrypted or not into.  Check is the correct error message is displaying for the invalid inputs.  Check is the SSL certificate is implemented or not.  Check on click of the back button of the browser the user should be logged out of the application.  Check is user is able to login by directly by entering the URL in the browser or not.  Check the login session timeout functionality.  Check the password field is encrypted or not.   1. What is the different between the STLC and SDLC? | | |  |
| **Parameters** | **SDLC** | **STLC** |
| Definition | Software Development Life Cycle is a process that mainly focuses on developing software that can meet the client’s expectations and work efficiently in the technological infrastructure. Besides, it is a cost-effective process. | Software Testing Life Cycle specifies what type of test activities must be carried out and when should the testing team accomplish those test activities. |
| Relationship | SDLC is taken as the predecessor. | STLC is taken as the successor. |
| Focus | The focus of SDLC is on both complete software development and the testing process. | The focus of STLC is only on the testing process. |
| Phases | The SDLC phases are Requirement Gathering, System Analysis, Software Designing, Coding, Software Test Execution, and Deployment & maintenance. | The STLC phases are Requirement Analysis, Test Planning, Designing of Test, Environment Setup, Software Test Execution, and Test Cycle Closure. |
| Requirement Analysis Phase | In SDLC, the business analyst collects all the requirements and creates a software development plan. | In STLC, the QA team analyses requirements and creates a perfect system test plan. |
| Design Phase | The SDLC team creates the high and low-level designs as per the project’s requirements. | The test lead or test architect  creates the test strategy. |
| Coding Phase | In the coding phase, the software developers create actual code. | The STLC testing team prepares for the test environment setup and executes test cases. |
| Testing Phase | Here the actual testing process starts which includes testing types like Unit testing, Integration testing, Regression testing, performance testing, etc. | Actual testing in this phase starts with defect reporting & retesting is done here. |
| Deployment or Maintenance Phase | In this last SDLC stage, the development team offers support and release updates. | The QA team executes regression suites to check the deployment of maintenance code. |

|  |  |  |
| --- | --- | --- |
| 1. 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. Help people outside the test team such as developers, business managers, customers understand the details of testing.  Test Plan guides our thinking. It is like a rule book, which needs to be followed  Important aspects like test estimation, test scope, Test Strategy are documented in Test Plan, so it can be reviewed by Management Team and re-used for other projects.  Analyze the product Design the Test Strategy  Define the Test Objectives Define Test Criteria  Resource Planning Plan Test Environment  Schedule & Estimation Determine Test Deliverables   1. what is priority?   Priority is defined as the order in which the defects should be resolved. The priority status is usually set by the testing team while raising the defect against the dev team mentioning the timeframe to fix the defect. The Priority status is set based on end users requirement.  For example: If the company logo is incorrectly placed in the company's web page then the priority is high but it is of low severity.  Priority means how fast it has to be fixed.  Priority is related to scheduling to resolve the problem.  Severity means how severe it is affecting the functionality.  Is largely related to Business or Marketing aspect. It is a pointer towards the importance of the bug.  The priority status is set based on the customer requirements.  Is related to technical aspect of the product. It reflects on how bad the bug is for the system.  Priority means how urgently the issue can be fixed.  Product manager is to decide the Priority to fix a bug.  Based on ‘Project Priorities the product fixes are done.  The Priority status is set by the tester to the developer mentioning the time frame to fix a defect. If High priority is mentioned then the developer has to fix it at the earliest.   1. What is 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.  It is totally related to the quality standard or devotion to standard.  Severity means how severe it is affecting the functionality.  Severity is associated with standards.  The severity type is defined by the tester based on the written test cases and functionality.  Is related to technical aspect of the product. It reflects on how bad the bug is for the system.  It is totally related to the quality standard or devotion to standard.  Severity means how big functionality is affecting of the product.  The Test Engineer can decide the severity level of the bug.  Based on Bug Severity the product fixes are done.  Also we can say The Severity status is used to explain how badly the deviation is affecting the build.   1. Bug categories are…   Classification of bugs in software testing is done on the basis of their nature and impact on the user experience.  Performance Bugs  Security Bugs  Unit Level Bugs  Functional Bugs  Usability Bugs  Syntax Errors  Compatibility Errors  Logic Bugs   1. Advantage of bugzilla.   There are many bug tracking systems at present. A bug tracker is an inherent part of any web site testing, desktop testing, or mobile testing. Each bug tracking instrument has its characteristics, strengths, and weaknesses.  it is an open-source widely used bug tracker;  it is easy in usage and its user interface is understandable for people without technical knowledge;  it easily integrates with test management instruments;  it integrates with an e-mailing system;  it automates documentation.  Not all testers like to work with Bugzilla. Some of them find that its interface is too sophisticated, complain that it is hard to manage the logged in errors and that there are many defects in the code of this instrument.   1. What are the different methodology in agile development model?   Agile refers to the methods and best practices for organizing projects based on the values and principles documented in the Agile Manifesto. However, there’s no one right way to implement Agile and many different types of methodologies from which to choose. Here are some of the most common Agile frameworks.  Kanban  Kanban is a simple, visual means of managing projects that enables teams to see the progress so far and what’s coming up next. Kanban projects are primarily managed through a Kanban board, which segments tasks into three columns: “To Do,” “Doing,” and “Done.”  Scrum  Scrum is similar to Kanban in many ways. Scrum typically uses a Scrum board, similar to a Kanban board, and groups tasks into columns based on progress. Unlike Kanban, Scrum focuses on breaking a project down into sprints and only planning and managing one sprint at a time. Scrum also has unique project roles: Scrum master and product owner.  Extreme Programming (XP)  Extreme Programming (XP) was designed for Agile software development projects. It focuses on continuous development and customer delivery and uses intervals or sprints, similar to a Scrum methodology. However, XP also has 12 supporting processes specific to the world of software development  Planning game Small releases  Customer acceptance tests Simple design  Pair programming Test-driven development  Refactoring Continuous integration  Collective code ownership Coding standard  Feature-driven development (FDD)  Feature-driven development is another software-specific Agile framework. This methodology involves creating software models every two weeks and requires a development and design plan for every model feature. It has more rigorous documentation requirements than XP, so it’s better for teams with advanced design and planning abilities. FDD breaks projects down into five basic activities:  Develop an overall model Build a feature list  Plan by feature Build by feature  Dynamic Systems Development Method (DSDM)  The Dynamic Systems Development Method (DSDM) was born of the need for a common industry framework for rapid software delivery. Rework is to be expected, and any development changes that occur must be reversible. Like Scrum, XP, and FDD, DSDM uses sprints. This framework is based on eight fundamental principles:  Focus on the business need Deliver on time  Collaborate Never compromise quality  Crystal is a family of Agile methodologies that includes Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, etc. Each has a unique framework. Your choice depends on several project factors, such as your team size, priorities, and project criticality.  Lean  Lean development is often grouped with Agile, but it’s an entirely different methodology that happens to share many of the same values. The main principles of the Lean methodology include:   1. Write a scenario of only whatsapp chat message.   Check the Chat window that contains the entire chat list.  Check the Chat window displays the contact numbers whose numbers are not saved on mobile.  Check the Chat window displayed with all contacts with DP or without DP  Check the Chat window is displayed on the group chat list.  Check the Chat window displays the last updated chatting time.  Check the Chat window displays the name of all contacts on the chat window.  Check the clicking on one Chat contact then a new window should open with history.  Check the user can see all delivered and received messages.  Check the user can see the read or send time of messages.  Check the user can send and receive text messages in the individual chatbox.  Check the user can send and receive documents in the individual chatbox.  Check the user can send and receive photos in an individual chatbox.  Check the user can send and receive videos in an individual chatbox.  Check the user can send and receive audio in an individual chat box.  Check the user can send and receive emotions icons in the individual chat boxes.  Check the user can send and receive Contacts in the individual chat boxes.  Check the user can send and receive Location in the individual chatbox.  Check the user can send and receive GIFs in the individual chat boxes.  Check the user can send and receive Stickers in the individual chatboxes.  Check the user can delete text, video, audio, locations, and documents in the individual chatboxes.  Check the user can send recorded voice mail in an individual chatbox.  Check the user can delete the entire chat history in the individual chatbox.  Check the user is able to see contact details in the individual chat box.  Check the user is able to share images, links, and documents from media in the individual chatboxes.  Check the user is able to search specific chat history using the search option in the individual chatbox.  Check the user is able to video call in the individual chat box.  Check the user is able to voice call in the individual chat box.  Check the user is able to mute the individuals in the individual chat boxes.  Check the user is able to change the wallpaper.  Check the users have options like Report, Block, Clear Chat, Export Chat, and Add Shortcut.  Test Scenarios For WhatsApp Group Chats  Check whether the user is able to create a new one or not.  Check the user is able to add multiple contacts from the contact list.  Check the user is able to insert the group name and select an image for DP.  Check the user is able to add and remove contacts from the group.  Check the user is able to delete a group.  Check the user can send and receive text messages in the group.  Check the user can send and receive documents in the group chat box.  Check the user can send and receive photos in the group chat box.  Check the user can send and receive videos in the group chat box.  Check the user can send and receive audio in the group chat box.  Check the user can send and receive emotions icons in the group chat box.  Check the user can send and receive Contacts in the group chat box.  Check the user can send and receive Location in the group chat box.  Check the user can send and receive GIFs in the group chat box.  Check the user can send and receive Stickers in the group chat box.  Check the user can delete text, video, audio, locations, and documents in the group chat box.  Check the user can send recorded voice mail in the group chat box.  Check the user is able to make multiple video call in the group chat box.  Check the user is able to see the group contact information from Group Info in the group chat box.  Check the user is able to share images, links, and documents from Group Media in the group chat box.  Check the user is able to search specific chat history using the search option in the group chat box.  Check the user is able to mute the group in the group chat box.  Check the users have options like Report, Block, Clear Chat, Export Chat, and Add Shortcut.   1. Write a scenario of pen   Verify if you are able to hold the pen comfortably.  Writing: Verify if you are able to write smoothly.  Verify that the pen is not making any sound while writing.  Verify the ink flow. It should not overflow nor get a break either.  Verify the quality of the material used for the pen.  Verify if the company or pen name is visible clearly.  Verify if the pen color or text written on the pen is not getting removed easily.  Verify, whether the width of the line drawn by the pen is as per the expectations or not.  Verify the ink color, it should be consistent from the start till the end.  Verify if a pen can write on a variety of papers like smooth, rough, thick, thin, glossy etc.  Verify for the waterproof ink. [Not for gel and ink pens].  Verify if the ink will not get dried easily by keeping the pen open for some time. [Not for ink pen]  Verify if any other refill fits in the pen or not.  Verify that the pen doesn’t have sharp edges or corners.  Verify if the ink and external assembly of the pen is made of non-toxic material.  **Nagative**  Verify if you are able to hold the pen comfortably.  Writing: Verify if you are able to write smoothly.  Verify that the pen is not making any sound while writing.  Verify the ink flow. It should not overflow nor get a break either.  Verify the quality of the material used for the pen.  Verify if the company or pen name is visible clearly.  Verify if the pen color or text written on the pen is not getting removed easily.  Verify, whether the width of the line drawn by the pen is as per the expectations or not.  Verify the ink color, it should be consistent from the start till the end.  Verify if a pen can write on a variety of papers like smooth, rough, thick, thin, glossy etc.  Verify for the waterproof ink. [Not for gel and ink pens].  Verify if the ink will not get dried easily by keeping the pen open for some time. [Not for ink pen]  Verify if any other refill fits in the pen or not.  Verify that the pen doesn’t have sharp edges or corners.  Verify if the ink and external assembly of the pen is made of non-toxic material.   1. Write a scenario of door   Verify if the door is single door or 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  Test the swiping with a Valid Card.  Test the swiping with Invalid Card.  Test the swiping of the Valid Card and get entry.  Test the swiping of the Card with both entry and exit with a Valid Card.  Test the swiping of the Card with Invalid Card continuously swiping and after limited attempts try to pop up the message that Card is blocked.  Test for Swiping of Card with the proper position in front of the machine.  Test for swiping of Card with proper ID and displaying Valid ID for reports extracted.  Test for swiping of Card with proper Time and Date and correct Id display for the reports extracted.  Test for swiping of Card with One or more Card swiping at the same time and should not allow multiple entries of overlapping of reports.  Test without using a Card and manually entering and manipulating the machine.  Test for swiping of Card and automatically close the door within time.  Test for swiping of Valid Card and message displayed as Access allowed or Thank You.  Test for swiping of Invalid Card and message displayed as Access denied.  Test for swiping of Invalid Card and again manually entering and manipulating the machine.  Test for swiping of In-Valid Card and have some advanced features if someone tries to enter and pop up the message immediately to the concerned department.   1. Scenario ATM   Verify if the card reader is working correctly. A screen should ask you to insert the pin after inserting the valid card.  Verify if the cash dispenser is working as expected.  Verify if the receipt printer is working correctly. Which means it can print the data on the paper and the paper comes out properly.  Verify if the Screen buttons are working correctly. For touch screen: Verify if it is operational and working as per the expectations.  Verify if the text on the screen button is visible clearly.  Verify the font of the text on the screen buttons.  Verify each number button on the Keypad.  Verify the functionality of the Cancel button on the Keypad.  Verify the text color of the keypad buttons. The numbers should be visible clearly.  Verify the text color and font of the data on the screen. The user should be able to read it clearly.  Verify the language selection option. If the messages or data are displayed in the selected language.  Insert the card, the correct pin, and print the receipt for available balance.  Verify the receipt printing functionality after a valid transaction. Whether the printed data is correct or not.  Verify how much time the system takes to log out.  Verify the timeout session functionality.  Verify the deposit slot functionality depending on its capability (Cash or cheque or both) by inserting a valid cheque.  Verify using different cards (Cards of different banks).  Verifying the Message  Insert the card and an incorrect PIN to verify the message.  Verify the message when there is no cash in the ATM.  Verify the messages after a transaction.  Verify if a user will get a correct message if a card is inserted incorrectly.  Verify the cash withdrawal functionality by inserting some valid amount.  Verify if a user can perform only one cash withdrawal transaction per PIN insert.  Verify the different combinations of operation and check if there will be a power loss in the middle of the operation.  **Negative**  Verify the functionality by entering a wrong pin number for 3 or more times.  Verify the card reader functionality by inserting an expired card.  Verify the deposit slot functionality by inserting an invalid cheque.  Verify the cash withdrawal functionality by inserting invalid numbers like 10, 20, 50 etc.  Verify the cash withdrawal functionality by entering an amount greater than the per day limit,  Verify the cash withdrawal functionality by entering an amount greater than per transaction limit.  Verify the cash withdrawal functionality by entering an amount greater than the available balance in the account.   1. When to used usability testing?   Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:  Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.  Time required to get used to in using the software.  The measure of increase in user productivity if any.   1. What is procedure for GUI testing?   GUI Testing is a process of testing the application’s graphical user interface to ensure proper functionality as per the specifications. It involves checking the application components like buttons, icons, checkboxes, color, menu, windows etc.  Visual dynamics of a web application play a pivotal role in the acceptance of an application with the user.  Consequently, this acceptance yields in bringing a long time bondage of customers with the client’s application. In this era of digitization, User Interface is fast changing and holds a key fortress in attracting the new crowd of possible customers.   1. Microwave owen scenario   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.   1. Coffee vending machine scenario   Verify that type of the coffee vending machine  Check that the company name of coffee vending machine  Check that the company logo is properly displayed or  Check the design of the vending machine  Check that material of the vending machine  Check that dimension of the vending machine  Check that color of the coffee vending machine  Check the size of the vending machine  Check the height of the vending machine  Check that weight of the coffee vending machine  Check that labels are properly displayed on buttons or not  Verify that all the buttons should be displayed properly.  Verify that coffee vending machine should be off when the user press on power OFF button  Verify that vending machine all buttons should be working properly  Verify that when the vending machine starts, the indicator lights should be working properly.  Verify that the mechanism should be working properly when ingredients are under capacity level  Verify that the auto cleaner facility is working properly or not  Verify that the water level indicator should be working properly.  Verify that the half-cup facility is working properly or not  Verify that the cup quantity counter should work properly.  Verify that the automatic temperature is working properly or not  Verify that the safety lock system is available or not.  Verify that the cleaner should work properly for the coffee vending machine.  **Nagative**  Verify that behavior of the coffee vending machine when the power supply is improper  Verify that coffee should not be leakage from anywhere on the machine  Verify that coffee vending machine sound should not be noisy.  Verify that coffee vending machine operation should be working properly if the water container’s capacity is exceeded.  Verify that vending machine operation should be working properly if the coffee container’s capacity is exceeded.  Verify that vending machine operation should be working properly if the milk container’s capacity is exceeded.  Verify that machine operation should be working properly if the sugar container’s capacity is exceeded.  Verify that coffee vending machine parts should not be damaged from anywhere on the machine.   1. Scenario of chair   Check the material used for making the chair is as per the requirement document.  Check if the dimension of the chair is as per the specification document.  Check if the dimension of the weight is as per the specification document.  Check if the dimension of the height is as per the specification document.  Check the number of legs of a chair.  Check the chair backrest option.  Check that all legs of the chair on a plane surface are equal or not.  Check if the chair is compatible for taking rest.  Check whether a human is able to sit comfortably or not on a chair.  Check if the chair has an adjustment functionality or not.  Check the sitting space as per mentioned in the requirement document.  Check the legs of the chair is having any wheels or not.  Check if the chair is good enough to handle some specified amount of load.  Check what is the maximum amount of load the chair is handling.  Check the date is stable enough to take any human load.  Check the color of the table is as per the SRS documents.  Check the type of chair, for example, Office chair, Dining room chair, Dentist chair, Beanbag chair, Swing chair, Public benches or Armchair.  **Negative Scenarios For Chair**  Check the balance of the chair with one arm.  Check the balance of the chair with three legs.  Check the stress testing of the chair by dropping the Chair down from the practical height.  Check there nothing is breaking, no damage to the Chair, and the Chair is performed without any issues.  Check how the Chair is working under different climate environmental conditions. |  |  |