**1>What is Exploratory Testing?**

**:-** Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution. It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.

The practice of exploratory testing has gathered momentum in recent years. Testers and QA managers are encouraged to include exploratory testing as part of a comprehensive test coverage strategy.

**2>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. It helps users identify which requirement produced the most number of defects during a testing cycle.

Not only does this show areas in need of improvement, but it also helps mitigate future roadblocks and identify process weaknesses.

**3>** **What is Boundary value testing?**

**:-** Functional testing is a type of software testing in which the system is tested against the functional requirements of the system. It is conducted to ensure that the requirements are properly satisfied by the application. Functional testing verifies that each function of the software application works in conformance with the requirement and specification. Boundary Value Analysis(BVA) is one of the functional testings.

Boundary testing is a black-box testing technique that software developers often use to check the errors at the boundaries or extreme ends of a given input domain. An input domain comprises all the possible inputs available in a software program. Software developers utilize black-box testing to analyze the behavior of a software program and examine its functionality. The extreme ends in boundary testing might include start to end, lower to upper or minimum to maximum. Instead of focusing only on the center of the data, boundary testing helps detect errors occurring at the boundary values of valid or invalid partitions.

**4>** **What is Equivalence partitioning testing?**

**:- Equivalence Partitioning Method** is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.   
  
**Guidelines for Equivalence Partitioning :**

* If the range condition is given as an input, then one valid and two invalid equivalence classes are defined.
* If a specific value is given as input, then one valid and two invalid equivalence classes are defined.
* If a member of set is given as an input, then one valid and one invalid equivalence class is defined.
* If Boolean no. is given as an input condition, then one valid and one invalid equivalence class is defined.

**5>** **What is Integration testing?**

**:- Integration testing** is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit-tested, integration testing is performed.

There are four types of integration testing approaches. Those approaches are the following:

1. **Big-Bang Integration Testing –** It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing. In simple words, all the modules of the system are simply put together and tested. This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during Big Bang integration testing is very expensive to fix.

**2. Bottom-Up Integration Testing –** In bottom-up testing, each module at lower levels are tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower-level modules.

**6>** **What determines the level of risk?**

**:-** Risk-reward is a general trade-off underlying nearly anything from which a return can be generated. Anytime you invest money into something, there is a risk, whether large or small, that you might not get your money back—that the investment may fail. For bearing that risk, you expect a return that compensates you for potential losses. In theory, the higher the risk, the more you should receive for holding the investment, and the lower the risk, the less you should receive, on average.**:-**

**7> What is Alpha testing?**

**:-** Alpha testing is a type of testing that is done on an application towards the end of a development process when the product is almost in a usable state.

This type of testing does not involve functional testing on the application. Instead, it is a user testing on the application in order to understand the user behavior and experience on the application. Normally this test is performed by test engineers, employees and sometimes friends / family members with the aim of trying to emulate around 80% of the customers. While these users test and give their feedback, the development team observes the behavior to check for design issues in the application. Alpha testing is mainly conducted to unveil bugs that might arise due to abrupt errors created by the users, validate the quality state of the software in minimal time and finally propound a build that procures the specifications required.

**8>** **What is beta testing?**

**:-** Beta testing is an opportunity for real users to use a product in a production environment to uncover any bugs or issues before a general release.

Beta testing is the final round of testing before releasing a product to a wide audience. The objective is to uncover as many bugs or usability issues as possible in this controlled setting.

Beta testers are “real” users and conduct their testing in a production environment running on the same hardware, networks, etc., as the final release. This also means it’s the first chance for full security and reliability testing because those tests can’t be conducted in a lab or stage environment.

**9> What is component testing?**

**:-** A software application is a combination of multiple components or modules. Testing the individual component of any application is called component testing. In this testing, the functionality and usability of each component are validated separately without integrating it with other components, also called module testing.

**10> What is functional system testing?**

**:-** Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user’s expectations. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

**11> What is Non-Functional Testing?**

**:-** Non functional testing is a type of software testing that verifies non functional aspects of the product, such as performance, stability, and usability. Whereas functional testing verifies whether or not the product does what it is supposed to, non functional testing verifies how well the product performs.

**Types of non-functional Testing**

1. Performance Tests
2. Load Tests
3. Stress Tests
4. Volume Tests
5. Security Tests
6. Upgrade & Installation Tests
7. Recovery Tests

**12> What is GUI Testing?**

**:-** GUI is the abbreviation of ‘Graphical User Interface’. It contains several visual elements, such as buttons, text boxes, menus, checkboxes, images, etc. GUI testing refers to the validating UI functions or features of an application that are visible to the users, and they should comply with business requirements. GUI testing is also known as UI testing. That means ‘User Interface testing. So, you can use both acronyms alternatively.

But why do we need GUI testing? GUI testing aims to ensure that the end-user gets a hassle-free experience. Since users are often unaware of the specific UIs, they focus on the design of the app, its colors, and whether it is easy to navigate. In this way, an app attracts more users day by day. For this reason, GUI testing became important.

**13> What is Adhoc testing?**

**:-** Adhoc testing is a type of software testing that is performed informally and randomly after the formal testing is completed to find any loophole in the system. For this reason, it is also known as Random or Monkey testing. Adhoc testing is not performed in a structured way so it is not based on any methodological approach. That’s why Adhoc testing is a type of Unstructured Software Testing.

**14> What is load testing?**

**:-**  load testing places a simulated “load” or demand on your web application to ensure it remains stable during operation. During a load test, testing software will measure the capacity of your web application via transaction response times. If your app features extended response times or becomes unstable at a certain level of simulated traffic, your software will have likely reached its peak operating capacity—which means a solution to this software bottleneck needs to be addressed and implemented.

**15> 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**.

**16> What is white box testing and list the types of white box testing?**

**:-** White Box Testing, or structural, code-based, or glass box testing, is a software testing technique that focuses on the software’s internal logic, structure, and coding. It provides testers with complete application knowledge, including access to source code and design documents, enabling them to inspect and verify the software’s inner workings, infrastructure, and integrations.

### Types of White Box Testing

* **Unit Testing:** Imagine you’re building a bicycle. [Unit testing](https://www.browserstack.com/guide/unit-testing-a-detailed-guide) would be like checking each part separately – testing the brakes, the gears, the pedals, etc., to ensure they all work correctly before assembling the whole bicycle.
* **Static Analysis:** This is like proofreading a book before it’s published. You’re looking for errors in grammar, punctuation, and sentence structure. Still, you need to read the book as a whole to understand the story (which would be more like dynamic analysis).
* **Dynamic Analysis:** This would be like test-driving a car. You’re not just looking at the components (like in static analysis), but you’re driving the car to see how it performs on the road.
* **Statement Coverage:** Imagine you’re a teacher checking a student’s homework. Statement coverage would be like ensuring the student has answered every question on the assignment.
* **Branch Testing:** This is like exploring all possible routes on a GPS. If you’re at an intersection, branch testing involves going straight, turning left, and turning right to ensure all paths lead to valid destinations.
* **Path Testing:** This would be like a postman ensuring they can deliver mail to every house on their route. They need to make sure every possible path is covered.
* **Loop Testing:** This is like checking a playlist on repeat. You want to ensure it loops back to the first song correctly after the last song finishes.

**17>**  **What is black box testing? What are the different black box testing techniques?**

:- Black box testing is a software testing method used to test a system without having any prior knowledge of the internal structure of the software under test. Instead, the focus is on how the system works as a whole. Specifically, this testing approach focuses on the input that goes into the software and the output that is produced to see whether the expected results are achieved.

## **Black Box Testing Techniques**

### 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.

### 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.

### 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.

**18> 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 affect software quality, software quality is some thing how smooth and reliable your software is. Smoothness and reliability is how less defects your software have. **Categories of defects:** Categories of defects are: Errors of commissions, Errors of omissions, Errors of clarity, and Error of speed and capacity.

**19> Mention 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.

**20> 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. These conditions are typically defined by engineering or test leadership to ensure quality standards are met.

Software testing teams will use exit criteria to determine if a test plan or project can exit to the next stage or be considered complete. This isn't something that should be left up to the subjective and/or ad hoc decisions of a test admin or SQA engineer, as it can directly impact the success of the next stage or project as a whole.

**21> When should "Regression Testing" be performed?**

**:-** Typically, regression testing is applied under these circumstances:

* A new requirement is added to an existing feature
* A new feature or functionality is added
* The codebase is fixed to solve defects
* The source code is optimized to improve performance
* Patch fixes are added
* A new version of the software is released
* When changes to the User Interface are made
* Changes in configuration
* A new third-party system is integrated with the current system

All of these occasions involve a restructuring or adjustment of the current code, which may result in unexpected behaviors, hence the need for regression testing.

**22> What is 7 key principles? Explain in detail?**

**:-** This tutorial introduces the seven basic Software Testing Principles that every Software tester and QA professional should know.

## 7 Principles of Software Testing

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy

## 1) Exhaustive testing is not possible

Yes! 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](https://www.guru99.com/defect-management-process.html)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.

## 4) Testing shows a presence of defects

Hence, testing principle states that – Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. [Software Testing](https://www.guru99.com/software-testing-introduction-importance.html) 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.

This leads us to our next principle, which states that- Absence of Error

## 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.

**23>** **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.

## **Quality control**‍

**Quality control**, alternatively, 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.

## **Testing**

**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.

Please check our ultimate guide for all the related QA questions: "

**24>**  **Difference between Smoke and Sanity?**

## **:- What Is Sanity Testing**

To understand sanity testing, let’s first understand software build. A software project usually consists of thousands of source code files. It is a complicated and time-consuming task to create an executable program from these source code files. The process to create an executable program uses “build” software and is called “Software Build”.

Sanity testing is performed to check if new module additions to an existing software build are working as expected and can pass to the next level of testing. It is a subset of regression testing and evaluates the quality of regressions made to the software.

## **What Is Smoke Testing**

Smoke Testing is carried out post software build in the early stages of SDLC (software development life cycle) to reveal failures, if any, in the pre-released version of a software. The testing ensures that all core functionalities of the program are working smoothly and cohesively. A similar test is performed on hardware devices to ensure they don’t release smoke when induced with a power supply. Thus, the test gets its name ‘smoke test’. It is a subset of acceptance testing and is normally used in tester acceptance testing, system testing, and integration testing.

The intent of smoke testing is not exhaustive testing but to eliminate errors in the core of the software. It detects errors in the preliminary stage so that no futile efforts are made in the later phases of the SDLC. The main benefit of smoke testing is that integration issues and other errors are detected, and insights are provided at an early stage, thus saving time.

**Differnace between Smoke Testing and Sanity Testing**

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke testing is done to assure that the acute functionalities of program is working fine. | Sanity testing is done to check the bugs have been fixed after the build. |
| Smoke testing is also called subset of acceptance testing. | Sanity testing is also called subset of regression testing. |
| Smoke testing is documented. | Sanity testing isn’t documented. |
| Smoke testing is performed by either developers or testers. | Sanity testing is normally performed by testers. |
| Smoke testing may be stable or unstable. | Sanity testing is stable. |
| Smoke testing is scripted. | Sanity testing is usually not scripted. |
| Smoke testing is done to measure the stability of the system/product by performing testing. | Sanity testing is done to measure the rationality of the system/product by performing testing. |
| Smoke testing is used to test all over function of the system/product. | Sanity testing is used in the case of only modified or defect functions of system/products. |
| Smoke testing can be performed either manually or by using automation tools. | Sanity testing is commonly executed manually, not by using any automation approach. |
| Smoke testing is performed when new product is built. | Sanity testing is conducted after the completion of regression testing. |
| It includes all the system’s essential basic functionality. | It includes only those modules where change in code is made. |
| Smoke Testing firstly performs on the initial build. Smoke testing is done first. | Sanity Testing is done on stable builds or for the introduced new features in the software. |
| Smoke testing can be carried out either way-manually or automatically. | Without using test cases or scripts sanity testing can be carried out. |
| There is end-to-end system verification done in smoke testing. | A specific component gets verified in sanity testing. |
| In the smoke testing process, the software build could be stable or unstable. | During sanity testing, the software build is comparatively stable. |
| For every new build release smoke testing is carried out. | Sanity testing is carried out when in-depth testing is not possible because of short time. |
|  |  |
|  |  |

**25>** **Difference between verification and Validation**

**:-**

|  |  |
| --- | --- |
| **verification** | **Validation** |
| Verification is the static testing. | Validation is the dynamic testing. |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| 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. |
| 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 finds about 50 to 60% of the defects. | Validation finds about 20 to 30% of the defects. |
| Verification is about process, standard and guideline. | Validation is about the product. |

**26>** **Explain types of Performance testing.**

**:-** Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measures according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

## **Types of Performance Testing for Software**

### Load Testing

Load testing measures system performance as the workload increases. That workload could mean concurrent users or transactions. The system is monitored to measure response time and system staying power as workload increases. That workload falls within the parameters of normal working conditions.

### Stress Testing

Unlike load testing, stress testing — also known as fatigue testing — is meant to measure system performance outside of the parameters of normal working conditions. The software is given more users or transactions that can be handled. The goal of stress testing is to measure the software stability. At what point does software fail, and how does the software recover from failure?

### Spike Testing

Spike testing is a type of stress testing that evaluates software performance when workloads are substantially increased quickly and repeatedly. The workload is beyond normal expectations for short amounts of time.

### Endurance Testing

Endurance testing — also known as soak testing — is an evaluation of how software performs with a normal workload over an extended amount of time. The goal of endurance testing is to check for system problems such as memory leaks. (A memory leak occurs when a system fails to release discarded memory. The memory leak can impair system performance or cause it to fail.)

### Scalability Testing

Scalability testing is used to determine if software is effectively handling increasing workloads. This can be determined by gradually adding to the user load or data volume while monitoring system performance. Also, the workload may stay at the same level while resources such as CPUs and memory are changed.

### Volume Testing

Volume testing determines how efficiently software performs with large projected amounts of data. It is also known as flood testing because the test floods the system with data.

**27>** **What is Error, Defect, Bug and failure?**

## **:-** **What is a Bug?**

A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which results in a bug. It is now that the Automation/ Manual Test Engineers describe this situation as a bug.

* A bug once detected can be reproduced with the help of standard bug-reporting templates.
* Major bugs are treated as prioritized and urgent especially when there is a risk of user dissatisfaction.
* The most common type of bug is a crash.
* Typos are also bugs that seem tiny but are capable of creating disastrous results.

## **What is a Defect?**

A defect refers to a situation when the application is not working as per the requirement and the actual and expected result of the application or software are not in sync with each other.

* The defect is an issue in application coding that can affect the whole program.
* It represents the efficiency and inability of the application to meet the criteria and prevent the software from performing the desired work.
* The defect can arise when a developer makes major or minor mistakes during the development phase.

## **What is an Error?**

Error is a situation that happens when the Development team or the developer fails to understand a requirement definition and hence that misunderstanding gets translated into buggy code. This situation is referred to as an Error and is mainly a term coined by the developers.

* Errors are generated due to wrong logic, syntax, or loop that can impact the end-user experience.
* It is calculated by differentiating between the expected results and the actual results.
* It raises due to several reasons like design issues, coding issues, or system specification issues and leads to issues in the application.

## **What is a Failure?**

Failure is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive. Generally, such situations happen very rarely because before releasing a product all possible scenarios and test cases for the code are simulated.  Failure is detected by end-users once they face a particular issue in the software.

* Failure can happen due to human errors or can also be caused intentionally in the system by an individual.
* It is a term that comes after the production stage of the software.
* It can be identified in the application when the defective part is executed.

**28>** **Difference between Priority and Severity**

**:-**

|  |  |
| --- | --- |
| **Priority** | **Severity** |
| Defined by the impact of a specific problem on any application’s functionality. | Defined by the impact on business. |
| Category decided by testers. | Category decided by developers or product owners. |
| Deals with the technical aspects of the application. | Deals with the timeframe or order to fix the defects. |
| The value does not change with time, it’s fixed. | The priority value is subjective and may change after comparing with other defects. |

**29>** **What is Bug Life Cycle?**

**:-** Software testing, despite the name, is much more than simply testing software. It is obviously important to identify as many bugs as possible, but finding the bugs is merely the beginning of the so-called **bug life cycle**. It is an ongoing battle to find, record, track, fix, and confirm that bugs are addressed. 

To be more specific, when a bug is found, it has to be recorded accurately and in such a way that the developer responsible for fixing it will be able to reproduce the problem. All of the bugs identified must be added to a database, tracked, assigned, fixed, and eventually closed. Communication must be clear, bugs must be well described and categorized so that management can assign them correctly, and every fix necessitates a verification process.

The complete life cycle of a bug can be lengthy. From identification through to a verified fix, the process requires the attention of testers, developers, project managers, and team leads. For large-scale testing projects, QA teams may even need a set of dedicated tools to manage bugs. All of this effort is highly rewarding for both the developing team and testing team in many respects:

* Allows both teams to approach the testing project (and especially bug management) in an unstructured manner
* Reduces the chance of overlooking critical issues
* Improves communication between team members and stakeholders
* Ensures that software quality is continuously monitored and enhanced
* Establishes a standard for the bug management process

### 10 Stages of a Bug Identification Workflow

The diagram below depicts a bug throughout its lifecycle. Let’s go through ten stages of the workflow:

**1. New:**This is the first stage in the life cycle of a bug. As a result, when a tester discovers a bug while testing applications, it falls into the ‘New’ category, and the bug is validated and tested in the subsequent stages of its life cycle.

**2. Assigned:**The bug is identified, approved by the testing lead, posted by the tester, and then assigned to the development team to work on. Finally, the testing team’s leader or QA manager assigns the bug to the developer.

**3. Active/Open:**During this phase, the developer analyzes the bug and devises a solution. Suppose the developer doesn’t believe the bug requires any fixing. In that case, they can assign the bug to one of the four remaining stages: Duplicate, Deferred, Rejected, or Not a Bug.

**4. Fixed:**After the developer analyzes the bug and makes the code changes to fix it, they can mark the bug as fixed and forward it to the testing team for further processing.

**5. Retest:**The tester retests the changed code, and the developer verifies with the testing team whether the specific bug has been fixed per the specified requirements.

**6. Closed:**This is the final stage of the bug life cycle. The tester retests it after the bug has been fixed. The tester changes the status from ‘Verified’ to ‘Closed’ if they believe no further code is required and the bug has been successfully resolved. The closed stage shows that the bug is free of defects.

**7. Rejected:**The bug is generally rejected if the developer believes the bug is inaccurate. The bug’s status then changes to ‘Rejected.’

**8.Duplicate:**The developer marks the status as ‘Duplicate’ if the same bug occurs again or if the concept of the bug matches the concept of another same bug.

**9.Deferred:**When a bug is marked as deferred, it is of lower priority and can be fixed in the next release. The deferred stage comprises several bug-related events, such as low priority, less time to fix, or a bug that cannot cause a major issue with the software product.

**10. Not a bug:**The status of a specific bug is marked as ‘Not a Bug’ when the application product has little or no changes. The bug does not affect the program’s functionality, which limits its performance.

**30>** **Explain the difference between Functional testing and Non-Functional testing**

**:-**

|  |  |
| --- | --- |
| **Functional testing** | **Non-Functional testing** |
| A functional requirement defines a system or its component. | A non-functional requirement defines the quality attribute of a software system. |
| It specifies “What should the software system do?” | It places constraints on “How should the software system fulfill the functional requirements?” |
| Functional requirement is specified by User. | Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers. |
| It is mandatory. | It is not mandatory. |
| It is captured in use case. | It is captured as a quality attribute. |
| Defined at a component level. | Applied to a system as a whole. |
| Helps you verify the functionality of the software. | Helps you to verify the performance of the software. |
| Functional Testing like System, Integration, End to End, API testing, etc are done. | Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done. |
| Usually easy to define. | Usually more difficult to define. |
| Example 1) Authentication of user whenever he/she logs into the system. 2) System shutdown in case of a cyber-attack. 3) A Verification email is sent to user whenever he/she registers for the first time on some software system. | Example 1) Emails should be sent with a latency of no greater than 12 hours from such an activity. 2) The processing of each request should be done within 10 seconds 3) The site should load in 3 seconds when the number of simultaneous users are > 10000 |

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

**:-**

|  |  |
| --- | --- |
| **STLC** | **SDLC** |
| SDLC is mainly related to software development. | STLC is mainly related to software testing. |
| Besides development other phases like testing is also included. | It focuses only on testing the software. |
| SDLC involves total six phases or steps. | STLC involves only five phases or steps. |
| In SDLC, more number of members (developers) are required for the whole process. | In STLC, less number of members (testers) are needed. |
| In SDLC, development team makes the plans and designs based on the requirements. | In STLC, testing team(Test Lead or Test Architect) makes the plans and designs. |
| Goal of SDLC is to complete successful development of software. | Goal of STLC is to complete successful testing of software. |
| It helps in developing good quality software. | It helps in making the software defects free. |
| SDLC phases are completed before the STLC phases. | STLC phases are performed after SDLC phases. |
| Post deployment support, enhancement, and update are to be included if necessary. | Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts. |
| Creation of reusable software systems is the end result of SDLC. | A tested software system is the end result of STLC. |

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

### :- Test Scripts

This story begins with the most detailed way to document testing, the test script. When people talk about test scripts, they usually mean a line-by-line description of all the actions and data needed to perform a test. A script typically has ‘steps’ that try to fully describe how to use the program — which buttons to press, and in which order — to carry out a particular action in the program. These scripts also include specific results that are expected for each step, such as observing a change in the UI. An example step might be “Click the ‘X’ button,” with an example result of “The window closes.”

When a tester first starts a new job, they might not know much about the product, the business domain, or even software testing. Scripts can help bridge that gap. If the tester carefully follows the directions — enter the string ‘abc’, click the submit button, make sure the form submitted and the value was saved — the test idea will be covered enough to consider it ‘tested’.

### Test Cases

The second most detailed way of documenting testing work is to use test cases. Test cases describe a specific idea that is to be tested, without detailing the exact steps to be taken or data to be used. For example, a test case might say “Test that discount codes can be applied on top of a sale price.” This doesn’t mention how to apply the code or whether there are multiple ways to apply the code. The actual testing that will cover this test case may vary from time to time. Will the tester use a link to apply a discount, or enter a code, or have a customer service rep apply the discount, or will they feel compelled to test every way to add a discount that they can think of? Test cases give flexibility to the tester to decide exactly how they want to complete the test.

This flexibility from test cases is both good and bad. Flexibility is beneficial when the tester is familiar with testing and familiar with the software under test and the current set of risks in the software. If the tester clearly understands what has already been tested, what has changed recently in the program, and how users typically use the program, they will choose an approach in their testing that will exercise both the most common user paths, and the less common paths that are most likely to reveal bugs.

On the other hand, if the tester does not have a good understanding of how the program is used, the recent risks to the program, and how to evaluate those risks as a tester, they may not have the information or skill they need to assess the actions required to reveal important bugs.

### Test Scenarios

The least detailed type of documentation is the test scenario. A test scenario is a description of an objective a user might face when using the program. An example might be “Test that the user can successfully log out by closing the program.” Typically, a test scenario will require testing in a few different ways to ensure the scenario has been satisfactorily covered. Just based on that light description, the tester might choose to close the program through the menu option, kill it through the task manager, turn the computer off, or see what happens when the program runs out of memory and crashes. Since test scenarios offer little information about how to complete the testing, they offer the maximum amount of flexibility to the tester responsible for them.

Like test cases, the flexibility that comes with using test scenarios creates similar benefits and drawbacks. Testing skill and domain knowledge make it easier for the tester to break test scenarios down into the relevant test ideas, select the approach that makes most sense, and perform tests that find important problems. This work is fun and challenging for a skilled tester, but it may be difficult or impossible for a novice unless they are able to collaborate with others to get the needed skill and perspective.

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

## **:- What is Test Plan**:

A test plan is a document that consists of all future testing-related activities. It is prepared at the project level and in general, it defines work products to be tested, how they will be tested, and test type distribution among the testers. Before starting testing there will be a test manager who will be preparing a test plan. In any company whenever a new project is taken up before the tester is involved in the testing the test manager of the team would prepare a test Plan.

* The test plan serves as the blueprint that changes according to the progressions in the project and stays current at all times.
* It serves as a base for conducting testing activities and coordinating activities among a QA team.
* It is shared with Business Analysts, Project Managers, and anyone associated with the project.

### ****Why is Test Plan creation important?****

* **Reduction in defects:** Identify and mitigate defects early on in the software lifecycle.
* **Improved quality:** Studies indicate that projects equipped with thorough test plans generally experience reduced development and maintenance expenses. Identifying and addressing issues in the early stages is more economical compared to dealing with them after the software release.
* **Resource utilization:** Data indicates that the presence of a test plan facilitates improved planning and allocation of resources, averting bottlenecks and ensuring the efficient execution of testing activities.
* **Faster time to market:**The implementation of effective testing strategies specified in the plan plays a role in streamlining the development process. This leads to quicker time to market.
* **Mitigate risks:** A well structured plan helps in identifying risks earlier in the process. These can be cross-collaborative risks as well as risks that are internal to the team.

### ****Who needs Test Planning?****

Test plan is an important artifact which, when well drafted, can act as an anchor for the product development process. However, these personas benefit more from its creation.

* **Developers,**who get insights into the testing scope and requirements. This helps them anticipate the tests that will be executed on their code and understand the success criteria for testing.
* **QAs,** who create and execute test cases to find bugs and document them. Test planning helps them establish testing strategies, allocate resources, and track the overall advancement of the testing process. Test Plan functions as a roadmap for carrying out test cases, monitoring test advancement, and guaranteeing thorough coverage.
* **Product Managers,** who use test plans to make informed decisions about release timelines, resource allocation, and overall product quality. It helps them coordinate release activities, assess customer impact, and foster effective collaboration between development and testing teams to deliver high-quality software.
* **Business Analysts,**who ensure all the test cases are well aligned with the business requirements specs at every stage, keeping users’ interests in mind. Through test planning, they get a clear vision of the relevance of test cases with current user requirements.
* **Compliance Teams**, ensure that the testing procedures conform to predefined standards as per regulatory demands.
* **Support Teams,**who communicate with users about known issues, provide timely solutions and offer guidance on workarounds. Test Plans help them anticipate potential issues or bugs that have been identified during the testing phase and figure out their workarounds to help users.

### 33> What are the different Methodologies in Agile Development Model?

**:-**

**What are agile Methodologies?**

Agile methodologies are systems of self-organization used by software development organizations to improve efficiency and encourage cooperation with the customer or intended software user. Software development teams often encounter issues they can fix by using these methodologies. These methodologies build teams of multi-disciplinary individuals to better meet deadlines and encourage continual improvement of the software.

**9 Common agile Methodologies**

### 1.Kanban

Kanban is a visual content management system used to implement constant delivery of developing work to clients. This method of workplace organization allows team members and clients to keep track of all aspects of the development process. This allows them to know where there are efficiency issues and actively provide solutions to ensure the team meets overall delivery timelines and customer satisfaction. Despite this improvement of efficiency, Kanban doesn't use individual timelines for each step of the project, so without proper oversight, there can be issues with project punctuality.

### 2.Scrum

### Scrum is a simple system of project organization that focuses on ownership of project items and due dates for each step of the process. Individual teams claim projects from a previously developed log of items that are necessary to complete. Priority items are labeled and completed first. Log items receive a “sprint” timeline designation which is a relatively short period during which they expect the team to complete their project. The team holds both daily and weekly meetings with a “scrum master” who handles general project oversight for the entire log.

### 3. Feature-driven development (FDD)

Feature-driven development works on a two-week project schedule. After breaking each facet of the development process into various features, teams work on developing these features separately. This requires higher levels of planning than some other agile methodologies. This method focuses on product features and seeks to ensure that teams create stable software to improve customer satisfaction.

### 4. Behaviour-driven development (BDD)

### Behaviour -driven development is an agile methodology that emphasizes the inclusion of non-technical developers into the development process. This allows for a continual review of software functionality from a non-technical aspect that improves end-user function. By breaking down the walls of the highly technical software development process, BDD creates a workplace with increased collaboration and communication from a highly varied interdisciplinary team.

### 5. Lean development

Lean development began as the application of the principles of the "lean manufacturing" system to software development. An emphasis on simplicity and minimalism is the essence of the lean development agile method. This minimalism works to create extreme efficiency and fast completion of projects. What would be multiple-part tasks within other systems are broken down into simple components and then completed by individual specialized team members. This process depends upon a solid developmental system before creation and implementation.

### 6. Adaptive software development (ASD)

### Adaptive software development, as its name might indicate, focuses on adaptation and change throughout the development process. During the development process, problems continually arise. Rather than forming a rigid framework for this process that may have a hard time overcoming unforeseen issues, ASD plans to continually change, learn and shift focus. This allows teams to collaborate and learn from each other and from the individual problems that arise to continually develop a better workflow system and to efficiently complete projects.

### 7. Crystal

### Crystal is an umbrella term for a series of methodologies that encompasses varying team sizes. Crystal methodologies can work for teams of as small as eight to as large as 1,000 team members. The organization's general team size outlines each method. An emphasis on communication seeks to improve efficiency and quality. The extensive collaboration and discussion among team members foster an improved workflow.

### Extreme programming (XP)

### Another customer satisfaction-centered methodology, extreme programming seeks to provide simple, properly working software that is continually reviewed by the client to ensure it meets their needs. XP fosters an environment that encourages developers to accept criticism and demands for the reworking of previously finished aspects of a program. Productivity lies at the heart of the XP system, with a large and well-framed hierarchy within teams and managers to improve efficiency.

### 9. Dynamic systems development method (DSDM)

### The dynamic systems development method is an agile methodology that focuses on the business aspects of software development. With an emphasis on the rapid delivery of products, DSDM uses sprints to ensure timeliness. Like other methodologies, the DSDM plans on the continual editing of software throughout the development process. The DSDM seeks to improve customer satisfaction and development efficiency by working from a viewpoint that champions business principles rather than traditional software development principles.

### 34> Write a scenario of only Whatsapp chat messages

### :-

* **Verify that a user can successfully create a new group.**
* **Confirm that the group is immediately visible in the user’s chat list.**
* **Test adding multiple members to the group.**
* **Confirm that all added members receive an invitation and join successfully.**
* **Test adding a group description.**
* **Verify that any member of the group can change the group name and icon.**
* **Verify that the group admin can remove a member from the group.**
* **Confirm that the removed member no longer has access to the group.**
* **Test a group member voluntarily leaving the group.**
* **Confirm that the group updates for the remaining members.**
* **Verify that the group creator can generate an invitation link.**
* **Confirm that users joining via the link become members without further approval.**
* **Test sharing images, videos, and documents within the group.**
* **Verify that users can search for specific messages within the group chat.**
* **Verify that the group admin can transfer admin privileges to another group member.**
* **Confirm that the new admin has the necessary permissions.**
* **Confirm that group information and messages are synchronized across multiple devices linked to the same account.**

### 35> Write a Scenario of Pen

:-

**.** Verify that the maker’s brand and/or logo is readily visible on the pen.

* Verify the text available on the pen is is readable and apparent.
* Verify that the pens material is used as specified in the requirement document.
* Verify that the colour of the pens body is same as specified in the requirement document.
* Verify the button colour to ensure it matches the refill colour. (if the pen has a variety of colour buttons).
* Verify that the pens length and diameter are designed as specified in the requirement document.
* Verify the type of pen such as Ink pen, Ballpoint pen, Gel Pen etc.,
* Verify whether the user can properly write on a variety of papers.
* Verify whether the pen is able to write on surfaces like cardboard, rubber etc., other than paper.
* Verify the consistency of the colour of the pens ink.
* Verify the odor of the pens link
* Verify whether the pen is with cap or without cap.
* Verify the weight of the pen. It should be too heavy.
* Verify the pen’s outer body to ensure it is sturdy. It should not be breakable with ease.
* Verify to see whether the text written by the pen (not a gel or a ink pen) is water-resistant.
* Verify to see whether the user can change the refill of the pen easily incase of ballpoint and gel pens.
* Verify to see whether the user can refill the pen with all the supported ink types incase of an ink pen.
* Verify the mechanism for refilling the ink pen is simple to use.
* Verify the grip of the pen and make sure whether the user can hold the pen without discomfort
* Verify to see whether the pen is making any noise while writing.
* Verify the pen for sharp edges and corners to ensure that it is not dangerous.
* Verify there is no ink has overflowed or stopped flowing completely.
* Verify the line drawn by the pen is as expected in terms of width.
* Verify to see whether the ink will dry rapidly by keeping the pen open for a few moments.
* Verify the ink and body of the pen to see if they’re made of non-toxic materials.

### 36> Write a Scenario of Door

### :-

1. Verify if the door is single door or bi-folded door.
2. Check if the door opens inwards or outwards.
3. Verify that the dimension of the doors are as per the specifications.
4. Verify that the material used in the door body and its parts is as per the specifications.
5. Verify that colour of the door is as specified.
6. Verify if the door is sliding door or rotating door.
7. Check the position, quality and strength of hinges.
8. Check the type of locks in the door.
9. Check the number of locks in the door interior side or exterior side.

10. Verify if the door is having peek-hole or not.

11. Verify if the door is having stopper or not.

12. Verify if the door closes automatically or not – spring mechanism.

13. Verify if the door makes noise when opened or closed.

14. Check the door condition when used extensively with water.

15. Check the door condition in different climatic conditions- temperature, humidity etc.

16. Check the amount of force- pull or push required to open or close the door.

### 37> Write a Scenario of ATM

### :-

### 1. Verify the ‘ATM Card Insertion Slot’ is as per the specification 2. Verify the ATM machine accepts card and PIN details 3. Verify the error message by inserting a card incorrectly 4. Verify the error message by inserting an invalid card (Expired Card) 5. Verify the error message by entering an incorrect PIN 6. Verify that the user is asked to enter the PIN after inserting a valid ATM Card 7. Verify that PIN is encrypted 8. Verify that there is an action like blocking of card occurs when the total no. of incorrect PIN attempts get surpassed 9. Verify the user is allowed to do only one cash withdrawal transaction per PIN request 10. Verify the machine logs out of the user session immediately after successful withdrawal 11. Verify the message when there is no money in the ATM 12. Verify the language selection functionality 13. Verify the cash withdrawal functionality by entering some valid amount 14. Verify the cash withdrawal functionality by entering an amount less than 100 15. Verify the cash withdrawal functionality by entering an amount greater than the total available balance in the account. 16. Verify the cash withdrawal functionality by entering an amount greater than per day limit 17. Verify the user is allowed to enter the amount again in case the amount entered is not valid. A proper message should be displayed. 18. Verify the ATM machine successfully takes out the money. 19. Verify the ATM machine takes out the balance printout after the withdrawal 20. Verify the font of the text displayed in ATM screen 21. Verify the text on the screen buttons visible clearly. 22. Verify the functionality of all the buttons on the keypad 23. Verify the text on the buttons visible clearly. 24. Verify that touch of the ATM screen is smooth and operational 25. Verify the user is allowed to choose different account types like Savings, Current etc., 26. Verify the different combinations of operation and check if there will be an electricity loss in the middle of the operation. If there is an electricity loss in the middle of the transaction then the transaction should be marked as null and the amount shouldn’t be disclosed to others. 27. Verify the functionality of the cash dispenser 28. Verify the functionality of the receipt printer 29. Verify whether the printed data is correct or not in the receipt 30. Verify how much time the system takes to log out.

### 38> When to used Usability Testing?

**:-** Concept testing is crucial because it helps you predict the success or failure of a finished product early in the process. Whether you’re in the early stages of vetting new products, redesigning a marketing campaign, launching a new logo, or starting a loyalty program, consider testing your concept. By doing so, you’ll collect valuable customer insights and better predict whether or not you’re investing in something no one will use.

It’s always worth remembering that you are not your user. With consumer trends changing more rapidly, the risk of investing resources into developing an idea that’s already out-of-date is too costly. Instead, to build a successful product or experience, make sure that your ideas resonate with your current or prospective customer.

### 39> What is the procedure for GUI Testing?

**:-** GUI testing is a fascinating and somewhat contentious subject in the realm of software testing. When it comes to determining what Interface testing is, there is uncertainty. For example, some people might mix it up with UI testing.

Similarly, there could be some variation between GUI testing and [**End-to-end testing**](https://www.appsierra.com/blog/end-to-end-testing) issues, which is a cause for more misunderstandings. Finally, there is discussion as to whether GUI checking is something that you can do in the first place.

### ****GUI Testing Fundamentals****

GUI is a term for a “graphical user interface.” It represents a form of user interface that enables users to use graphical elements, such as keys, icons, and pictures, and often audio to communicate with electronic products or software.

To allow people to communicate with computers and other devices more easily than typing commands through a keyboard, GUIs were implemented. Generally, GUIs proved extremely effective and enabled the general, non-technical, community to popularise computing.

Today, not just on our screens, but on our tablets, our TVs, the dashboards in our vehicles, and many other locations, we will find GUIs.

Modern programs are either mobile-based or cloud-based software outside of the desktop. As per client demand, they have to be more user-friendly. The interface of the application and user experience play an important role in the performance of the application as it is introduced to the market. To achieve these goals, engaging the right **application software development services** is essential.

In visual dynamics, a GUI testing team often pays careful attention to each detail to ensure end-user satisfaction and ease.  
  
When it comes to ensuring the quality of your software, partnering with a reputable [**software test company**](https://www.appsierra.com/blog/top-10-software-testing-companies) can make all the difference.

### 40> Write a scenario of Microwave Owen

**:-**

1. Verify that the dimensions of the oven are as per the specification provided.
2. Verify that the oven’s material is optimal for its use as an oven and as per the specification.
3. Verify that the oven heats the food at the desired temperature properly.
4. Verify that the oven heats food at the desired temperature within a specified time duration.
5. Verify the ovens functioning with the maximum attainable temperature.
6. Verify the ovens functioning with minimum attainable temperature.
7. Verify that the oven’s plate rotation speed is optimal and not too high to spill the food kept over it.
8. Verify that the oven’s door gets closed properly.
9. Verify that the oven’s door opens smoothly.

10.Verify the battery requirement of the microwave oven and check that it function’s smoothly at that power.

11.Verify that the text written over the oven’s body is clearly readable.

12.Verify that the digital display is clearly visible and functions correctly.

13.Verify that the temperature regulator is smooth to operate.

14.Verify that the temperature regulator works correctly.

15.Check the maximum capacity of the oven and test its functioning with that volume of food.

16.Check the oven’s functionality with different kinds of food – solid, and liquid.

17.Check the oven’s functionality with different food at different temperatures.

18.Verify the oven’s functionality with different kinds of container material.

19.Verify that the power cord of the oven is long enough.

20.Verify that the usage instruction or user manuals have clear instructions.

### 41> Write a scenario of Coffee Vending Machine

### :-

* Verify whether the power button of the coffee vending machine is working correctly after pressing the power button.
* Verify whether the coffee vending machine is activated when the user presses the Power ON button.
* Verify whether the coffee vending machine is turned off when the user presses the power OFF button.
* Verify whether the indicator lights display correctly when the coffee vending machine is going to switch off or on.
* Verify whether all the buttons of the coffee vending machine have an image text on them, which indicates what task will be performed if you press the button.
* Verify whether the foamer in the coffee vending machine is working as expected.
* Verify whether the auto cleaner facility is working properly or not.
* Verify whether the half-cup feature works properly or not.
* Verify whether the cup quantity counter should work properly.
* Verify whether the temperature of the coffee served should be the same temperature or not.
* Verify whether the input mechanism for coffee ingredients-milk, water, coffee beans/powder, etc works as expected.
* Verify whether the quantity of hot water, milk, and coffee powder per serving are correct.
* Verify the effect of suddenly switching off the machine or cutting the power, the machine should stop in that situation and power resumption, the remaining coffee should not come out of the nozzle.
* Verify whether the functioning of all the buttons work properly when pressed
* Verify whether the coffee beans are grinding evenly, check it by picking a test bean and testing how evenly it has been ground.

### 42> Write a scenario of chair

### :-

1. Verify that the chair is stable enough to take an average human load.
2. Check the material used in making the chair-wood, plastic etc.
3. Check if the chair’s leg are level to the floor.
4. Check the usability of the chair as an office chair, normal household chair.
5. Check if there is back support in the chair.
6. Check if there is support for hands in the chair.
7. Verify the paint’s type and colour.
8. Verify if the chair’s material is brittle or not.
9. Check if cushion is provided with chair or not.

10.Check the condition when washed with water or effect of water on chair.

11.Verify that the dimension of chair is as per the specifications.

12.Verify that the weight of the chair is as per the specifications.

13.Check the height of the chair’s seat from floor.

### 43> Write a Scenario of Wrist Watch

### :-

1. Verify the type of watch – analog or digital.
2. In the case of an analog watch, check the correctness time displayed by the second, minute, and hour hand of the watch.
3. In the case of a digital watch, check the digital display for hours, minutes, and seconds is correctly displayed.
4. Verify the material of the watch and its strap.
5. Check if the shape of the dial is as per specification.
6. Verify the dimension of the watch is as per the specification.
7. Verify the weight of the watch.
8. Check if the watch is waterproof or not.
9. Verify that the numbers in the dial are clearly visible or not.

10.Check if the watch is having a date and day display or not.

11.Verify the colour of the text displayed in the watch – time, day, date, and other information.

12.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.

13.Check if the second hand of the watch makes ticking sound or not.

14.Verify if the brand of the watch and check if its visible in the dial.

15.Check if the clock is having stopwatch, timers, and alarm functionality or not.

16.In the case of a digital watch, verify the format of the watch 12 hours or 24 hours.

17.Verify if the watch comes with any guarantee or warranty.

18.Verify if the dial has glass covering or plastic, check if the material is breakable or not.

19.Verify if the dial’s glass/plastic is resistant to minor scratches or not.

20.Check the battery requirement of the watch.

### 44> Write a Scenario of Lift(Elevator)

### :-

* Check whether the controls/buttons in the lift are self-explanatory.
* Check whether the controls/ buttons are accessible and usable by a blind person.
* Check whether the voice guide technology is working as expected.
* Check whether the control pad is comfortable to use
* Check whether the experience inside a lift makes the user uncomfortable due to gravitational pull
* Check whether the lift is making noise while moving up or down.
* Check whether the floor number is being announced on each floor
* Check whether the lift provides light and aroma along with some instrumental music.
* Check whether the lift interior has proper air ventilation.
* Check whether the light of the button gets on when the user presses it.
* Check whether the light colour gets changed when a user reaches the destination floor.
* Check the time taken by the lift to reach every floor
* Check how much time the lift taken from the top of the building to the ground floor
* Check how many people lift can take regardless of their weight
* Check the behaviour of the lift when it takes more people than capacity.
* Check how many runs the lift makes on power backup.
* Check whether the lift indicates some alert message weight exceeds from standard.
* Check whether the lift doesn’t move to any floor when no button is clicked in the control panel.
* Check whether the lift stops at each floor when all the floors are clicked in the control panel.

### 45> Write a Scenario of WhatsApp Group (generate group)

**:-**

* Verify that the admin role can be assigned to a specific group member.
* Test the functionality of admin-specific permissions, such as adding/removing members and changing group settings.
* Check if admins receive notifications for important group activities, like member additions or removals.
* Verify that only admins can create, add/remove, and make other users admin in the group.
* Test the ability to add new members to a group, ensuring they receive invitations.
* Test the functionality of adding multiple members to a group simultaneously.
* Check if new members have appropriate permissions, such as sending messages and sharing media.
* Verify that existing members receive notifications for new member additions.
* Verify that users can create a new broadcast list.
* Test adding contacts to the broadcast list, ensuring accurate inclusion.
* Test the maximum limit of recipients in a broadcast list.
* Verify the system’s behaviour when attempting to exceed the limit.
* Confirm that users can edit the recipients in an existing broadcast list.
* Verify that users can delete a broadcast list.
* Test sending text messages, images, and documents via broadcast.
* Confirm that all recipients receive the broadcasted content.
* Verify that recipients can reply to a broadcast.
* Test if replies are visible only to the sender.
* Test the feature allowing users to forward broadcast messages to other contacts.