Module–2(Manual Testing)

1. What is Exploratory Testing?

* Exploratory testing is a type of manual testing that relies on testers’ experience and intuition, rather than scripts, to guide the testing process. It is especially good for catching issues that slip through the cracks with automated tests, for testing UI elements and for ensuring software actually meets its requirements. Exploratory testers are often guided by “charters,” which set out the ultimate goals for testing but do not tell the tester what to do or focus on.
* **Exploratory Testing** is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.
* Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

1. What is traceability matrix?

* A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.
* It is used to track the requirements and to check the current project requirements are met.
* RTM **Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

1. What is Boundary value testing?

* [Boundary Value Analysis](https://www.geeksforgeeks.org/boundary-value-analysis-triangle-problem/) is based on testing the boundary values of valid and invalid partitions. The behavior at the edge of the equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects.
* It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

1. What is Equivalence partitioning testing?

* **Equivalence Partitioning Method** is also known as Equivalence class partitioning (ECP). It is a [software testing](https://www.geeksforgeeks.org/software-testing-basics/) technique or [black-box testing](https://www.geeksforgeeks.org/software-engineering-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.

1. What is Integration testing?

* **Integration Testing** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated
* Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as **‘I & T’** (Integration and Testing), **‘String Testing’** and sometimes **‘Thread Testing’**.

1. What determines the level of risk?

* Determining the level of risk usually involves trying to assess not only the likelihood of an identified risk from actually occurring, but also the potential magnitude the consequences this risk could have on an organisation and its stakeholder, should it occur.
* **High**
* The impact of this risk would be very damaging and potentially non-tolerable. Ultimately, a risk of this scale could see the organisation make a loss should it occur. If a high risk is identified and cannot be solved, the software project may be too big of a risk to complete.
* **Medium**
* Problems, challenges or glitches labeled as ‘medium risk’ may be tolerable but are certainly not desirable. These risks may see financial loss in the short-term, however, if a solution can be found, the positives of finishing the software project will outweigh the negative risks.
* **Low**
* ‘Low’ risks can almost be classed as inconveniences or minor snags rather than actual threats to the project. Little to no financial loss would be seen in the event of one of these risks playing out

1. What is Alpha testing?

* Alpha testing is the first end-to-end testing of a product to ensure it meets the business requirements and functions correctly. It is typically performed by internal employees and conducted in a lab/stage environment. An alpha test ensures the product really works and does everything it's supposed to do.

1. what is beta testing?

* Beta testing software **evaluates product performance in the real world, prior to an official product launch, by obtaining feedback from a targeted group of users**.
* Beta testing is a form of user acceptance testing and takes place after alpha testing. Beta tests help to isolate any bugs and issues that may have been missed during earlier testing phases, which if left, could severely impact the quality and stability of the software.

1. What is component testing?

* Component testing, also known as **program or module testing**, is done after unit testing. In 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 like unit testing with the difference that the developer uses real data instead of dummy data for testing of the written code.  
    
  Suppose there is a software application which consists of five components modules. The testing of each module is done independently by the developer as part of the development cycle before it is ready for integration testing. By doing component testing, bugs can be found at a very early stage in the cycle and helps save time.  
    
  Debugging tools or unit test structure tools are used for this type of testing since this is done by programmers on the code written by them and with the support of integrated development environment. Defects are fixed as soon as possible when they are found without formally recording incidents.  
    
  Component testing plays an important role in finding the issue. Before we proceed with the integration testing it’s always advised to do the component testing in order to ensure that each module of an application is working correctly and as per requirement.

1. 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.
* At the end of [functional testing](https://www.microfocus.com/solutions/functional-testing-software-testing), you should have software that has a coherent user interface, a consistent API, and seamlessly integrates with business processes
* functional testing focuses on the results of processing and not the mechanics of the processing, and determines whether the application satisfies the basic minimum user expectations.

1. What is Non-Functional Testing?

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

* **Types of Non Functional Testing** :

1. Performance Testing
2. Load Testing
3. Failover Testing
4. Compatibility Testing
5. Usability Testing
6. Stress Testing
7. Maintainability Testing
8. Scalability Testing
9. Volume Testing
10. Security Testing
11. Disaster Recovery Testing
12. Compliance Testing
13. Portability Testing
14. Efficiency Testing
15. Reliability Testing
16. Baseline Testing
17. Endurance Testing
18. Documentation Testing
19. 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, etc.
* A user doesn’t have any knowledge about XYZ software/Application. It is the UI of the Application which decides that a user is going to use the Application further or not.
* A normal User first observes the design and looks of the Application/Software and how easy it is for him to understand the UI. If a user is not comfortable with the Interface or find Application complex to understand he would never going to use that Application Again. That’s why, GUI is a matter for concern, and proper testing should be carried out in order to make sure that GUI is free of Bugs.

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.
* Ad hoc 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.
* This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

1. What is load testing?

* **Load Testing** is a type of [Performance Testing](https://www.geeksforgeeks.org/performance-testing-software-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.
* Load testing is a type of performance testing that simulates a real-world load on a system or application to see how it performs under stress. The goal of load testing is to identify bottlenecks and determine the maximum number of users or transactions the system can handle. It is an important aspect of software testing as it helps ensure that the system can handle the expected usage levels and identify any potential issues before the system is deployed to production.
* During load testing, various scenarios are simulated to test the system’s behavior under different load conditions. This can include simulating a high number of concurrent users, simulating a large number of requests, and simulating heavy network traffic. The system’s performance is then measured and analyzed to identify any bottlenecks or issues that may occur.

### Some common load testing techniques include:

* **Stress testing:**Testing the system’s ability to handle a high load above normal usage levels
* **Spike testing:** Testing the system’s ability to handle sudden spikes in traffic
* **Soak testing**: Testing the system’s ability to handle a sustained load over a prolonged period of time
* Tools such as Apache JMeter, LoadRunner, Gatling, and Grinder can be used to simulate load and measure system performance. It’s important to ensure that the load testing is done in an environment that closely mirrors the production environment to get accurate results.

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***.

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.

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.

1. Mention what are the categories of defects.

* **Software Defect** is some kind of error, flaw or some kind of mistake from the development team which prevent the software from the smooth working. It directly 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.

1. 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.

1. What is the purpose of exit criteria?

* Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.
* Exit criterion should be part of test plan and decided in the planning stage.
* Verify if All tests planned have been run.
* Verify if the level of requirement coverage has been met.
* Verify if there are NO Critical or high severity defects that are left outstanding.
* Verify if all high risk areas are completely tested.
* Verify if software development activities are completed within the projected cost.
* Verify if software development activities are completed within the projected timelines.

1. When should "Regression Testing" be performed ?

* Regression testing should be performed after introducing new code changes to ensure they have not impacted the existing functionality of your system. This kind of testing can be done using manual and automated techniques
* **whenever changes are made to the system**, including new features, bug fixes, and performance improvements. It is an important part of the software development process and can help to ensure that changes made to a system do not introduce new bugs.

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

* **According to the ISTQB (International Software Testing Qualifications Board), the seven principles of software testing are:**

1. Testing shows the presence of defects.
2. Exhaustive testing is impossible.
3. Early testing.
4. Defect clustering.
5. Pesticide paradox.
6. Testing is context dependent.
7. Absence-of-errors fallacy.

## Testing shows the presence of defects

You test software to identify problems so you can fix them before you deploy the software to production environments. However, this process doesn't mean that there aren't any bugs in the product. It just means that there may be bugs, but you didn't find them.

There could be any number of reasons that you didn't uncover every bug, including the fact that the test cases didn't cover every scenario.

This principle, which helps to set stakeholder expectations, means that you shouldn't guarantee that the software is error-free.

## Exhaustive testing is impossible

The truth is that you can't test everything, i.e., every combination of preconditions and inputs. And if you try to do so you'll waste time and money, but it won't affect the overall quality of the software.

What you need to do is assess risk and plan your tests around these risks so you can be sure you're testing the key functions. Careful planning and assessment ensures your test coverage is good so you can have confidence in your final product — and you don't even have to test every individual line of code.

## Early testing

When it comes to the software development lifecycle, testing early is the key to identifying any defects in the requirements or design phase as soon as possible. It's much easier and less expensive to fix bugs in the early stages of testing than at the end of the software lifecycle as then you might have to rewrite entire areas of functionality. And that likely means missed deadlines and cost overruns.

## Defect clustering

Defect clustering is the idea that a small number of software modules or components contain the most defects — sort of applying the Pareto Principle to software testing, i.e., approximately 80% of the issues are found in 20% of the components.

Understanding this can help in your testing because if you find one defect in a particular area, you'll likely find more in that same module. If you identify the complex areas that are changing the most or the ones that have more dependencies, you can focus your testing on these key areas of risk.

## Pesticide paradox

This principle centers around the theory that if you repeatedly use a particular pesticide on your crops, the insects you're trying to kill or repel will eventually become immune to the pesticide and it will no longer be effective.

Likewise, if you continuously run the same tests, eventually they'll fail to find new defects, even though they'll probably confirm the software is working.

Consequently, you must continue to review your tests as well as add to your scenarios or modify them to help prevent this pesticide paradox. For example, maybe you could use a variety of testing techniques, methods, and approaches simultaneously.

Testing is context dependent

Software testing is all about the context, which means that no one strategy will fit every scenario. The types of testing and the methods you use totally depend on the context of the systems or the software, e.g., the testing of an iOS application is different from the testing of an e-commerce website. Put simply, what you're testing will always affect the approach you use.

## Absence-of-errors fallacy

If your software is 99% error-free but it doesn't follow your user's requirements, it's still not usable. That's why it's critical to run tests that pertain to the requirements of the system. Software testing isn't just about finding bugs, it's about ensuring that the software meets the user's needs and requirements.

As such, you should also test your software with the users. You can test against early prototypes at the usability testing phase so you can get feedback from the users that you can use to ensure the software is usable. Even though your software might have relatively few issues, doesn't mean it is ready to ship; it also has to meet your customer's requirements and expectations.

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

* QA, QC, and Testing are like a chain that works together to ensure a high-quality product. QA sets the standards for how it should be done, QC ensures those standards are followed, and Testing checks that everything is up to the quality standards.
* **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.
* 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.

1. Difference between Smoke and Sanity.

* The goal of Smoke testing is to verify stability, whereas the goal of Sanity testing is to verify rationality. Software Developers or Testers perform smoke testing, whereas testers alone perform sanity testing. Smoke testing is a subset of acceptance testing, while sanity testing is a subset of regression testing.
* The purpose of smoke testing is to verify the critical functionalities of a system, while sanity testing verifies the new functionality such as bug fixes
* Smoke testing is a subset of acceptance testing, while sanity testing is a subset of regression testing.
* Smoke testing is documented or scripted, while sanity testing is not.
* In smoke testing, the entire system is verified from end to end. In sanity testing, on the other hand, only a particular component of the system gets verified.
* Smoke test is done to make sure that the critical functionalities of the program are working fine, whereas sanity testing is done to check that newly added functionalities, bugs, etc., have been fixed.
* The software build may be either stable or unstable during smoke testing. The software build is relatively stable at the time of sanity testing.
* Smoke testing is done on initial builds, while sanity testing is done on relatively stable builds
* Smoke testing is done as a part of basic testing, whereas sanity testing is done as part of regression testing.
* Smoke testing is usually done every time there is a new build release. But sanity testing is planned when there is not sufficient time for in-depth testing.
* Smoke testing is like a general health checkup, while sanity testing resembles a specialized health checkup.

1. Difference between verification and Validation.

| **Verification** | **Validation** |
| --- | --- |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does *not* include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |
| It consists of checking of documents/files and is performed by human. | It consists of execution of program and is performed by computer. |

1. Explain types of Performance testing.

* Stress Testing
* This test pushes an application beyond normal load conditions to determine which components fail first. Stress testing attempts to find the breaking point of the application and is used to evaluate the robustness of the application’s data processing capabilities and response to high volumes of traffic.
* Spike Testing
* This testing evaluates the ability of the application to handle sudden volume increases. It is done by suddenly increasing the load generated by a very large number of users. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load. This testing is critical for applications that experience large increases in number of users; for example, utility customers reporting power outages during storms. This can be considered a component of stress testing
* Load Testing
* The purpose of load testing is to evaluate the application’s performance under increasingly high numbers of users. Load, or increasing numbers of users are applied to the application under https://qualitestgroup.com/initiatives/load-and-performance-testing-services/test and the results are measured to validate the requirements are met. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.
* Endurance Testing
* Endurance testing evaluates the performance of the system under load over time. It is executed by applying varying loads to the application under test for an extended period of time to validate that the performance requirements related to production loads and durations of those loads are met. Endurance testing can be considered a component of load testing and is also known as soak testing.
* Volume Testing
* Also known as flood testing, this testing is used to evaluate the application’s ability to handle large volumes of data. The impact on response time and the behavior of the application are analyzed. This testing can be used to identify bottlenecks and to determine the capacity of the system. This type of performance testing is important for applications that deal with big data.
* Scalability Testing
* This testing is used to determine your application’s ability to handle increasing amounts of load and processing. It involves measuring attributes including response time, throughput, hits and requests per second, transaction processing speed, CPU usage, Network usage and more. Results of this testing can be used in the planning and design phases of development which reduces costs and mitigates the potential for performance issues.

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

* Testing is the process of identifying defects, where a defect is any variance between actual and expected results. “A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure.”

1. Difference between Priority and Severity.

* Severity is basically a parameter that denotes the total impact of a given defect on any software. Priority is basically a parameter that decides the order in which we should fix the defects. Severity relates to the standards of quality. Priority relates to the scheduling of defects to resolve them in software.

1. What is Bug Life Cycle?

* **Defect Life Cycle** or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.
* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

1. Explain the difference between Functional testing and Non-functional testing.

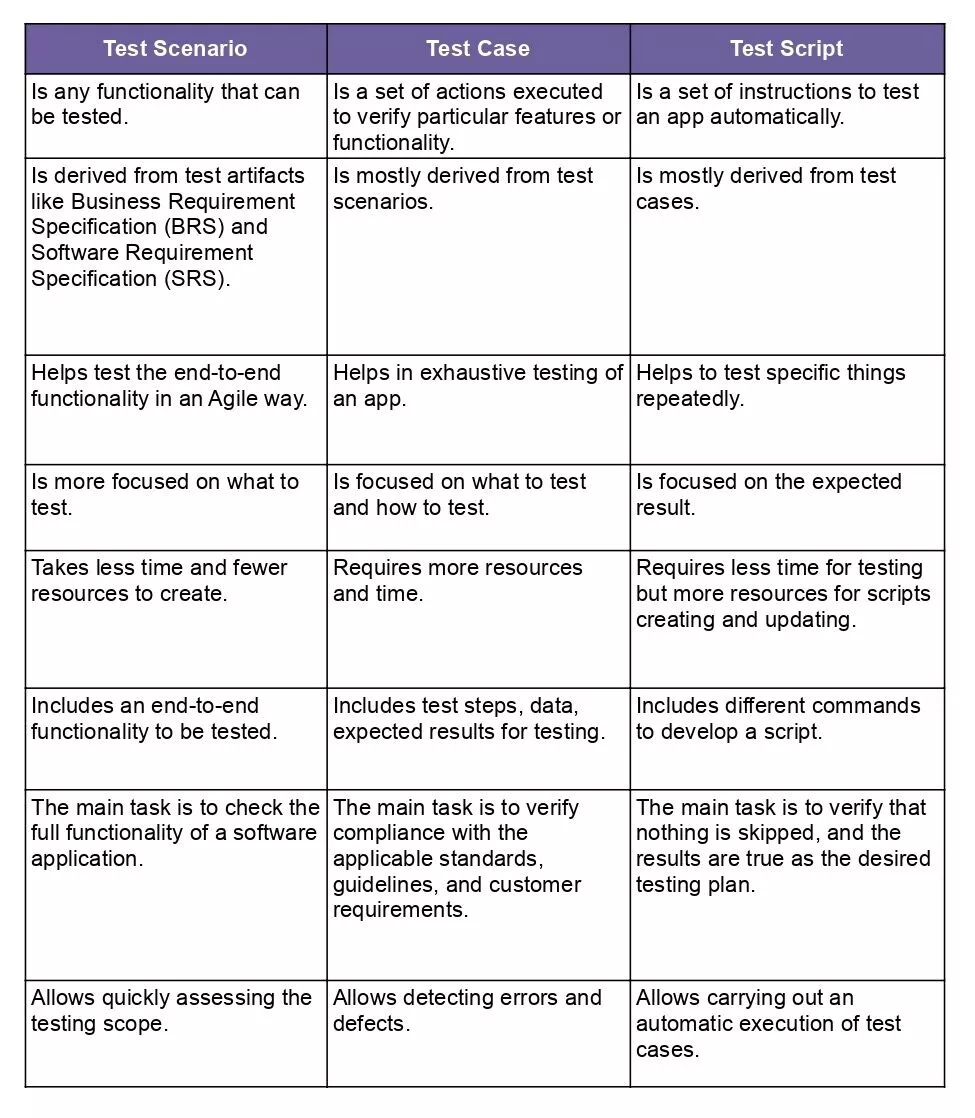


| **Functional Testing** | **Non-functional Testing** |
| --- | --- |
| It verifies the operations and actions of an application. | It verifies the behavior of an application. |
| It is based on requirements of customer. | It is based on expectations of customer. |
| It helps to enhance the behavior of the application. | It helps to improve the performance of the application. |
| Functional testing is easy to execute manually. | It is hard to execute non-functional testing manually. |
| It tests what the product does. | It describes how the product does. |
| Functional testing is based on the business requirement. | Non-functional testing is based on the performance requirement. |
| **Examples:**  **1.** Unit Testing  **2.** Smoke Testing  **3.** Integration Testing  **4.** Regression Testing | **Examples:**  **1.** Performance Testing  **2.** Load Testing  **3.** Stress Testing  **4.** Scalability Testing |

1. What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?
2. **Difference between SDLC and STLC:**

| **SDLC** | **STLC** |
| --- | --- |
| 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. |

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

* A test scenario is any functionality that a software testing company can examine. It is also called a Test Condition or Test Possibility. A test case is a document that lists the steps a QA engineer needs to execute. A test script is a short program written in a programming language. It is used to test a part of the functionality of a software

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

* A Test Plan is a detailed document that catalogs the test strategies, objectives, schedule, estimations, deadlines, and resources required to complete that project. Think of it as a blueprint for running the tests needed to ensure the software is working correctly – controlled by test managers.
* Scope of Testing
* This section is important to include because it lists the overall areas included and excluded from testing. It provides an insight into the approved scope of work for the QA team and works as an excellent reference for reporting.
* Features to be tested – Include a list of all the functionality to be tested. Providing a reference to the requirement is always helpful as it’s useful in tracking back and gathering overall coverage information.
* Features not to be tested – Include a list of all the elements which will not be part of the Verification. Do be sure to include a reason for exclusion. Common scenarios like payment related testing, access limitation to the system, location constraints, and cases bound by the security protocol, all fall out of scope for QA team.
* Testing Approach
* The primary objective of a test plan is to communicate the testing approach/methods to the reader. This section must list the types and number of rounds of testing, the defect tracking process and list of tools to be used for the project.
* Item pass/fail criteria
* We all know the general rule that when the system matches the requirement, the test is considered a pass, but during testing conditions can arise when the testers aren’t sure whether to mark something as pass or fail. Identify such criteria and include them in the test plan. A common example of this would be if an event was unreproducible or concerned environmentally related defects.
* Suspension criteria and resumption requirements
* During the test execution, there may be instances when the testing needs to be suspended for a period. The test plan must explicitly mention the condition when test activity has to stop and when it can resume. A few examples to consider could be “Testing will be suspended when 40% or more of the defects are open,” or ” Testing will be suspended if blocked defects have not been fixed.”
* Test deliverables
* Test deliverables must give a list of all the documents planned for creation during the QA cycle. Along with it, mention the schedule and timeline for the production and submission of these documents.

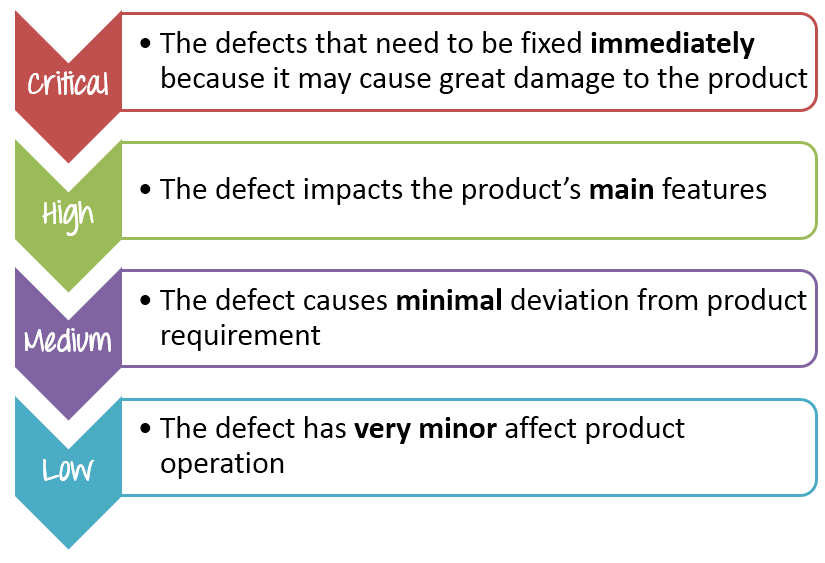
1. What is priority ?

* 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.

1. What is severity?

* One can define Severity as the extent to which any given defect can affect/ impact a particular software. Severity is basically a parameter that denotes the impact of any defect and its implication on a software's functionality. In other words, Severity defines the overall impact that any defect can have on a system.

1. Bug categories are…

* bugs can be classified as urgent, high-, medium-, and low-priority.
* 

1. Advantage of Bugzila

* The Advantages of Bugzilla are:
* 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.

1. Difference between priority and severity.

* Severity is basically a parameter that denotes the total impact of a given defect on any software. Priority is basically a parameter that decides the order in which we should fix the defects. Severity relates to the standards of quality. Priority relates to the scheduling of defects to resolve them in software.

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

* There are 5 main Agile methodologies: **Scrum, Kanban, Extreme Programming (XP), Lean Development e Crystal**.

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

* Authentication verifies the identity of a user or service, and authorization determines their access rights. Although the two terms sound alike, they play separate but equally essential roles in securing applications and data. Understanding the difference is crucial. Combined, they determine the security of a system.
* **4 Key Software Testing Challenges**
* Lack of Communication.
* Missing Documentation.
* Diversity in Testing Environments.
* Inadequate Testing.

1. Write a scenario of only Whatsapp chat messages.
2. Verify that ‘Chats’ window contains all the chat list with DP and name and last message preview of the other person with whom chat was initiated.
3. Verify that clicking a chat in the chat list opens a new window containing all the chats received and sent with the other person.
4. Verify that the user can check the message delivered and read the time for a message in the ‘Message Info’ section.
5. Verify that the user can share or receive contact with the other person.
6. Verify that the user can create a group by adding multiple people from his contact list.
7. Verify that the user can send and receive the message in group chats.
8. Verify that users can send and receive images, audio, video, and emoticons in the chat with individuals.
9. Verify that users can send and receive images, audio, video, and emoticons in group chats.
10. Verify that the user can send and receive chats in the secondary languages available.
11. Verify that users can delete text, images, audio, and video messages within a chat.
12. Verify that users can clear their complete chat history in an individual or group chat.
13. Verify that users can archive chats in an individual or group chat.
14. Verify that users can block a user to prevent any message from getting received from the blocked
15. Write a Scenario of Pen.
16. **The grip of the pen:** Verify if you are able to hold the pen comfortably.
17. **Writing:** Verify if you are able to write smoothly.
18. Verify that the pen is not making any sound while writing.
19. Verify the ink flow. It should not overflow nor get a break either.
20. Verify the quality of the material used for the pen.
21. Verify if the company or pen name is visible clearly.
22. Verify if the pen color or text written on the pen is not getting removed easily.
23. Verify, whether the width of the line drawn by the pen is as per the expectations or not.
24. Verify the ink color, it should be consistent from the start till the end.
25. Verify if a pen can write on a variety of papers like smooth, rough, thick, thin, glossy etc.
26. Verify for the waterproof ink. [Not for gel and ink pens].
27. Verify if the ink will not get dried easily by keeping the pen open for some time. [Not for ink pen]
28. Verify if any other refill fits in the pen or not.
29. Verify that the pen doesn’t have sharp edges or corners.
30. Verify if the ink and external assembly of the pen is made of non-toxic material.
31. Write a Scenario of Pen-drive .

* Data gets stored (i.e., written to)in a pen drive from a magnetic drive like HDD, Optical drives like CD-ROM, and other storage devices.
* Written data can be retrieved (read-back operation)
* The device can be detected across its interface. Say, if the pen drive is a USB device, it is detected in a USB drive.
* The detected pen drive reflects the correct capacity, it claims.
* Data written to or read from the pen drive should not be corrupted.
* The pen drive can be formatted correctly and should be able to host the file system with which it is formatted.

1. Write a Scenario of Door
2. 1. Verify if the door is single door or bi-folded door
3. Check if the door opens inwards or outwards
4. Verify that the dimension of the doors are as per the specifications
5. Verify that the material used in the door body and its parts is as per the specifications
6. Verify that color of the door is as specified
7. Verify if the door is sliding door or rotating door
8. Check the position, quality and strength of hinges
9. Check the type of locks in the door
10. Check the number of locks in the door interior side or exterior side
11. Verify if the door is having peek-hole or not
12. Verify if the door is having stopper or not
13. Verify if the door closes automatically or not – spring mechanism
14. Verify if the door makes noise when opened or closed
15. Check the door condition when used extensively with water
16. Check the door condition in different climatic conditions- temperature, humidity etc
17. Check the amount of force- pull or push required to open or close the door
18. Write a Scenario of ATM.

* Test Cases for 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.

1. When to used Usablity Testing?

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

1. What is the procedure for GUI Testing?

* A test plan defines the scope of the test project. Before running test cases, it is important to [create a test plan](https://reqtest.com/testing-blog/software-test-plan-template-2/) to identify the scope of the project, resources available and functionalities to be tested in the application. The entire team works on creating [test scenarios](https://reqtest.com/testing-blog/test-scenario-test-case/), creating test cases and scripts to start with testing.
* A test script is a template which defines the standard of input/ information required to test the GUI of the application. It includes:
* Test Script ID: It uniquely identifies the test script.
* Title: It defines the Test Script, the part of the functionality under test.
* Test Case ID: Unique ID which is used to link it with test cases.
* Test Setup: Defines all the requirements of the test environment.
* Test Data: Data values that are used to check the usability and correctness of the application.
* Procedure: A series of steps used to define the instructions of the test. The entire process of GUI testing.
* Priority/Risk Level: Assigning a risk level to the test case (critical, high, medium, low).
* Description: The entire information about the test case.
* Expected Result: It defines success as the expected result.
* Status: It defines the status of the test case.
* Creating test scripts is an important part of GUI testing. Any customization in the test scripts fields to suit business needs and demands is a great advantage of any tool. [ReQtest](https://reqtest.com/try-reqtest/) is one such tool which allows you to customize the test scripts, manage and streamlines test cases to improve testing speed and quality of the application. It supports [agile methodology](https://reqtest.com/agile-blog/agile-methodology-tutorial/) and can provide great assistance to the entire team.

1. Write a scenario of Microwave Owen.
2. Verify that the dimensions of the oven are as per the specification provided.
3. Verify that the oven’s material is optimal for its use as an oven and as per the specification.
4. Verify that the oven heats the food at the desired temperature properly.
5. Verify that oven heats food at the desired temperature within a specified time duration.
6. Verify the ovens functioning with maximum attainable temperature.
7. Verify the ovens functioning with minimum attainable temperature.
8. Verify that the oven’s plate rotation is speed is optimal and not too high to spill the food kept over it.
9. Verify that the oven’s door gets closed properly.
10. Verify that the oven’s door opens smoothly.
11. Verify the battery requirement of the microwave oven and check that it function’s smoothly at that power.
12. Verify that the text written over the oven’s body is clearly readable.
13. Verify that the digital display is clearly visible and functions correctly.
14. Verify that the temperature regulator is smooth to operate.
15. Verify that the temperature regulator works correctly.
16. Check the maximum capacity of the oven and test its functioning with that volume of food.
17. Check oven’s functionality with different kinds of food – solid, liquid.
18. Write a scenario of Coffee vending Machine.

 UI scenario – Verify that the dimension of the coffee machine is as per the specification

 Verify that outer body, as well as inner part’s material, is as per the specification

 Verify that the machine’s body color as well brand is correctly visible and as per specification

 Verify the input mechanism for coffee ingredients-milk, water, coffee beans/powder, etc

 Verify that the quantity of hot water, milk, coffee powder per serving is correct

 Verify the power/voltage requirements of the machine

 Verify the effect of suddenly switching off the machine or cutting the power. The machine should stop in that situation and in power resumption, the remaining coffee should not get come out of the nozzle.

 Verify that coffee should not leak when not in operation

 Verify the amount of coffee served in single-serving is as per specification

 Verify that the digital display displays correct information

 Check if the machine can be switched on and off using the power buttons

 Check for the indicator lights when the machine is switched on-off

 Verify that the functioning of all the buttons work properly when pressed

 Verify that each button has an image/text with it, indicating the task it performs

 Verify that complete quantity of coffee should get poured in a single operation, no residual coffee should be present in the nozzle

 Verify the mechanism to clean the system work correctly- foamer

 Verify that the coffee served has the same and correct temperature each time it is served by the machine

 Verify that system should display an error when it runs out of ingredients

 Verify that pressing the coffee button multiple times leads to multiple serving of coffee

 Verify that there is the passage for residual/extra coffee in the machine

 Verify that machine should work correctly in different climatic, moistures and temperature conditions

 Verify that machine should not make too much sound when in operation

1. Write a scenario of chair.

 Verify that the chair is stable enough to take an average human load

 Check the material used in making the chair-wood, plastic etc

 Check if the chair’s leg are level to the floor

 Check the usability of the chair as an office chair, normal household chair

 Check if there is back support in the chair

 Check if there is support for hands in the chair

 Verify the paint’s type and color

 Verify if the chair’s material is brittle or not

 Check if cushion is provided with chair or not

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

 Verify that the dimension of chair is as per the specifications

 Verify that the weight of the chair is as per the specifications

 Check the height of the chair’s seat from floor

1. To Create Scenario facebook Chat on Mobile
   1. Check received messages counts should be displayed on 'Facebook Message' icon

2. Verify that user gets all received messages in his inbox

3. Verify that only 'message contacts' will display in left hand side of message box

4. Verify that profile picture display in left hand side of inbox is correct for each user

5. Verify that 'Active' users display with green dot in message box

6. Verify that unread messages are highlighted so that user can identify it

7. Check received messages counts should be displayed with Inbox in 'Messages' page

8. Verify that messages will get display in Inbox of 'User1' only when 'sender' is connected with user1 on Facebook

9. Verify that messages will get display in 'Other' tab of 'User1' if 'sender' is not connected with user1 on Facebook

11. Verify that user can search contacts in message box

12. Check behavior of chat box if we change network from Wi-Fi to LAN

13. Verify that user is able to navigate to old conversation or can view message history

14. Verify that user is able to send new message to friend selected from list present at left hand side

15. Verify that message get sent after clicking on enter button

16. Verify that copy, paste works in chat box or not

17. Verify that the User is able to send special characters in Chat or not.

18. Verify that the User is able to share hyperlinked URLs, Emails, or not.

19. Verify that how many words or characters can be sent at a time.

20. Verify that spell functionality works fine in chat box

21. Check if user enters message in textbox and click on refresh button without sending it

22. Verify that user is able to send smiley

23. Verify that user is able to send multiple smiles at a time

24. Verify that if user types smiles in letters then it will look like their icon or not

25. Verify that the User is able to share images

26. Verify that error message should get display after uploading image of unsupported type

27. Verify that the User is able to share videos

28. Verify that the User is able to share files

29. Verify error message should get display after uploading large size files

30. Verify that user is able to send messages in local languages

31. Verify that if user has typed any message and navigated to another tab without sending it then message should not get removed

32. Verify that user gets appropriate message if internet goes down while sending message

33. Verify that user is able to view that which device has been used to send a message

1. Write a Scenario of Wrist Watch.

 Verify the type of watch – analog or digital.

 In the case of an analog watch, check the correctness time displayed by the second, minute, and hour hand of the watch.

 In the case of a digital watch, check the digital display for hours, minutes, and seconds is correctly displayed.

 Verify the material of the watch and its strap.

 Check if the shape of the dial is as per specification.

 Verify the dimension of the watch is as per the specification.

 Verify the weight of the watch.

 Check if the watch is waterproof or not.

 Verify that the numbers in the dial are clearly visible or not.

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

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

 Verify that clock’s time can be corrected using the key in case of an analog clock and buttons in case of a digital clock.

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

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

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

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

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

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

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

 Check the battery requirement of the watch.

1. Write a Scenario of Lift(Elevator)

 Verify the dimensions of the lift

 Verify the type of door of the lift is as per the specification

 Verify the type of metal used in the lift interior and exterior

 Verify the capacity of the lift in terms of the total weight

 Verify the buttons in the lift to close and open the door and numbers as per the number of floors

 Verify that lift moves to the particular floor as the button of the floor is clicked

 Verify that lift stops when up/down buttons at particular floor are pressed

 Verify if there is an emergency button to contact officials in case of any mishap

 Verify the performance of the floor – the time is taken to go to a floor

 Verify that in case of power failure, lift doesn’t free-fall and get halted in the particular floor

 Verify lifts working in case button to open the door is pressed before reaching the destination floor

 Verify that in case door is about to close and an object is placed between the doors if the doors sense the object and again open or not

 Verify the time duration for which door remain open by default

 Verify if lift interior is having proper air ventilation

 Verify lighting in the lift

 Verify that at no point lifts door should open while in motion

 Verify that in case of power loss, there should be a backup mechanism to safely get into a floor or a backup power supply

 Verify that in case multiple floor number button is clicked, lift should stop at each floor

 Verify that in case of capacity limit is reached users are prompted with warning alert- audio/visual

 Verify that inside lift user are prompted with current floor and direction information the lift is moving towards- audio/visual prompt