# Module-2 And 3 ASSIGNMENTS (Manual Testing)

**What is exploratory testing?**

* Exploratory testing is a concurrent process where test design, execution and logging happen simultaneously.
* Testing is not often recorded.
* Testing is based on a test charter that may include scope of testing (in and out).
* The focus of exploratory testing is more on testing as a thinking activity.
* It is not a technique but it is an approach. what action you perform is governed by what you are doing currently

**What is traceability matrix?**

* To protect against changes, you should be able to trace back from every system component to the original requirement that caused its presence
* There are basically three types of traceability matrix

1) Forward traceability: mapping of requirement to test cases

2) Backward traceability: mapping of test cases to requirements

3) bi-directional traceability: a good traceability matrix is the reference from test cases to basis documentation and vice versa

**What is boundary value testing?**

* Boundary value analysis technique used to check boundary of the input
* Boundary value analysis focuses on boundary rather than value
* Boundary value analysis is a methodology for designing test cases that concentrates software testing effort on cases near

The limit of valid ranges

* Boundary value analysis generates test cases that highlight errors better than equivalence partitioning

**What is equivalence partitioning testing?**

* Portion data into various classes and we can select data according to class then test it reduces the number of test cases and save time for testing
* Equivalence partitioning technique focuses on value enter by the user

**What is integration testing?**

* Integration testing is defined as a type of testing where software module is integrated logically and tested as a group
* The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated
* Integrated testing focuses on checking data communication amongst these module

**What determines the level of risk?**

* A Risk could be any future event with a negative consequence Need to identify the risks associated with your project.
* When testing does find defects, the Quality of the software system increases when those defects are fixed
* A properly designed test that passes, reduces the overall level of Risk in a system.

**What is alpha testing?**

* **Alpha Testing** is a type of software testing performed bugs before releasing to identify the software product to the real users or public.
* The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.
* This testing is referred to as an alpha testing only because it is done early on, near the end of the development of the software, and before Beta Testing
* Alpha testing is typically performed by in-house software engineers or QA staff. It is the final testing stage before the software is released into the real world.

**What is beta testing?**

* Beta testing is performed by real user of the software application and it can be considered as a form of external user acceptance testing it is final test before shipping a product to the customer
* Direct feedback from customer is major advantage of beta testing. this testing helps to test product in customer environment
* Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

**What is component testing?**

* Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It is also referred to as Module Testing when it is viewed from an architecture perspective.
* Component Testing is also referred to as Unit Testing, Program testing or Module Testing.
* Generally, any software is made of several components. Component Level Testing deals with testing these components individually.
* It is one of most frequent black box testing types which is performed by QA Team
* Component testing is performed soon after the Unit testing is done by the developers and the build is released for the testing team. This build is referred as UT builds (Unit Testing Build). Major functionality of all the components is tested in this phase.

**What is functional system testing?**

* A requirement that specifies a function that a system or system component must perform
* There is two type of test approach

1) Requirement based functional testing

2) Process based testing

* Testing against requirements and specifications
* Test procedure and cases derived from detailed user requirements

**What is nonfunctional system testing?**

* Testing the attribute of a component or system that do not relate to functionality e.g., reliability, efficiency, usability, interoperability, maintainability, and portability
* It is testing of how the system works
* The term nonfunctional testing describes the tests required to measure characteristics of systems and software that can be quantified on a varying scale such as response times for performance testing
* To address this issue performance testing is carried out to check system response time the goal of performance testing is to reduce response time to an acceptable level

**What is GUI testing?**

* Gui 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
* GUI is what the user sees. Say if you visit facebook.com what you will see say homepage it is the GUI (graphical user interface) of the site
* A user does not see the source code. The interface is visible to the user. Especially the focus is on the design structure, images that they are working properly or not.

**What is adhoc testing?**

* An adhoc 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.

**What is load testing?**

* **Load Testing** is a non-functional software testing process in which the performance of software application is tested under a specific expected load
* It determines how the software application behaves while being accessed by multiple users simultaneously
* The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.
* Loading testing identifies the following problems before moving the application to market or Production
* Response time for each transaction
* Performance of System components under various loads
* Performance of Database components under different loads
* Network delay between the client and the server
* Software design issues
* Server configuration issues like a Web server, application server, database server etc.
* Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

**What is stress testing?**

* **Stress Testing** is a type of software testing that verifies stability & reliability of software application
* The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software does not crash under crunch situations
* It even tests beyond normal operating points and evaluates how software works under extreme conditions.
* A most prominent use **of stress testing is to determine the limit, at which the system or software or hardware breaks**
* It also checks whether the system demonstrates effective error management under extreme conditions

**What is white box testing and list the type 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.
* Types of white box testing
* **Unit testing** — tests written as part of the application code, which test that each component is working as expected.
* **Integration testing** — tests specifically designed to check integration points between internal components in a software system, or integrations with external systems.

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

* **Black-box testing:** Testing, either functional or non-functional, without reference to the internal structure of the component or system.
* Specification-based testing technique is also known as ‘black-box’ or input/output driven testing techniques because they view the software as a black-box with inputs and outputs.
* The testers have no knowledge of how the system or component is structured inside the box. In black-box testing the tester is concentrating on what the software does, not how it does it.
* **There are four different black box testing techniques:**
* Equivalence partitioning
* Boundary value analysis
* Decision tables
* State transition testing

**Mention what are the categories of defects?**

* A software bug arises when the expected result does not match with the actual results. It can also be error, flaw, failure, or fault in a computer program. Most bugs arise from mistakes and errors made by developers, architects.
* **Category of Defects:**

1. **Data Quality/Database Defects:** Deals with improper handling of data in the database. **Examples:**

Values not deleted/inserted into the database properly

Improper/wrong/null values inserted in place of the actual values

1. **Critical Functionality Defects:** The occurrence of these bugs hampers the crucial functionality of the application. **Examples: - Exceptions**
2. **Functionality Defects:** These defects affect the functionality of the application.

**Examples:**

All JavaScript errors

Buttons like Save, Delete, Cancel not performing

their intended functions

A missing functionality (or) a feature not

functioning the way, it is intended to Continuous execution of loops

1. **Security Defects:** Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority fora fix.

**Examples:**

Authentication: Accepting an invalid username/password

Authorization: Accessibility to pages though permission not Given

1. **User Interface Defects:** As the name suggests, the bugs deal with problems related to UI are usually considered less severe.

**Examples:**

Improper error/warning/UI messages

Spelling mistakes Alignment problems

**Mention what bigbang testing is?**

* In Big Bang integration testing all components or modules is integrated simultaneously, after which everything is tested.
* Big Bang testing has the advantage that everything is finished before integration testing starts.
* The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.
* Here all component is integrated together at once, and then tested.

**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 the test activities right from planning, specification, and execution.
* Exit criterion should be part of test plan and decided in the planning stage

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

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

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

* **7 keys principles are:**
* Testing shows presence of Defects.
* Exhaustive Testing is Impossible.
* Early Testing.
* Defect Clustering.
* The Pesticide Paradox.
* Testing is Context Dependent.
* Absence of Errors Fallacy.

1. **Testing shows presence of Defects:**

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

1. **Exhaustive Testing is Impossible:**
   * Testing everything including all combinations of inputs and preconditions is not possible.
   * So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
2. **Early Testing:**

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

1. **Defect Clustering:**

* A small number of modules contain most of the defects discovered during pre- release testing, or are responsible for the most operational failures.
* Defects are not evenly spread in a system they are ‘clustered
* In other words, most defects found during testing are usually Confined to a small number of modules similarly, most operational failures of a system are usually confined
* To a small number of modules
* An important consideration in test prioritization!

1. **The Pesticide Paradox:**

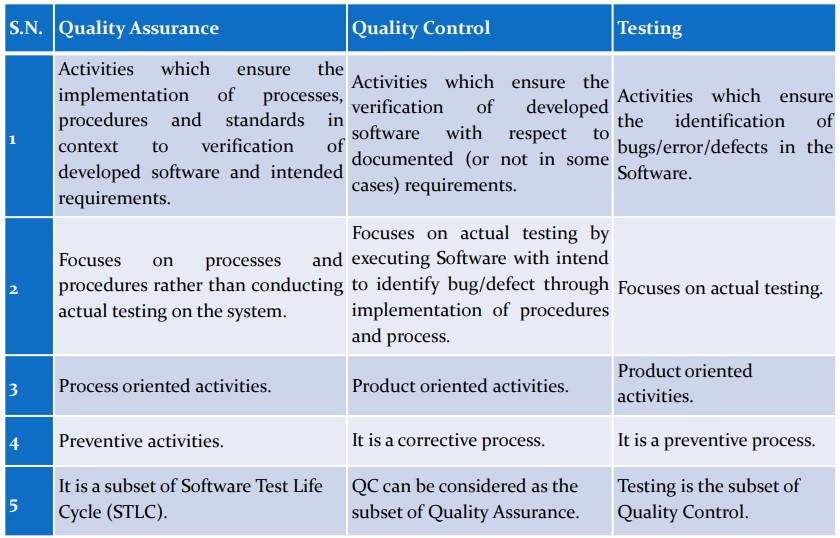
* If the same tests are repeated overland over again Eventually the same set of test cases will no longer find any new defects.
* To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
* Testing identifies bugs, and programmers respond to fix them.
* As bugs are eliminated by the programmers, the software improves
* As software improves the effectiveness of previous testserodes Therefore, we must learn, create, and use new tests based on new Techniques to catch new bugs.

1. **Testing is Context Dependent:**

* Testing is basically context dependents.
* Testing is done differently in different contexts.
* Different kinds of sites are tested differently.
* For example
* Safety – critical software is tested differently from an E-commerce site.

1. **Absence of Errors Fallacy:**

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



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

**Difference between Smoke and Sanity?**

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

**Difference between verification and Validation**

|  |  |
| --- | --- |
| verification | validation |
| The verifying process includes checking documents, design, code, and program | It is a dynamic mechanism of testing and validating the actual product |
| It does not involve executing the code | It always involves executing the code |
| Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc. | It uses methods like Black Box Testing, White Box Testing, and non-functional testing |
| Whether the software conforms to specification is checked | It checks whether the software meets the requirements and expectations of a customer |
| It finds bugs early in the development cycle | It can find bugs that the verification process cannot catch |
| Target is application and software architecture, specification, complete design, high level, and database design etc. | Target is an actual product |
| QA team does verification and make sure that the software is as per the requirement in the SRS document. | With the involvement of testing team validation is executed on software code. It comes before validation |
| It comes before validation | It comes after verification |

**Explain types of Performance testing.**

* **Types of Performance Testing:**
* Load testing
* Stress testing
* Endurance testing
* Spike testing
* Volume testing
* Scalability testing

1. **Load testing:**

* Load testing - It is a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.
* Load testing is a kind of performance testing which determines a system’s performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.
* This testing usually identifies –
* The maximum operating capacity of an application
* Determine whether current infrastructure is sufficient to run the application
* Sustainability of application with respect to peak user load
* Number of concurrent users that an application can support, and scalability to allow more users to access it.
* It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.

1. **Stress testing:**

* Stress testing - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
* Stress testing is used to test the stability & reliability of the system. This test mainly determines the system on its robustness and error handling under extremely heavy load conditions.
* It even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
* Stress Testing is done to make sure that the system would not crash under crunch situations.
* Stress testing is also known as endurance testing.
* Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity.
* Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.

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

* “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”
* Error: A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.
* Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.
* Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.
* Fault: An incorrect step, process, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner. See: bug, defect, error, exception.
* Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

**Difference between Priority and Severity**

|  |  |
| --- | --- |
| Priority | Severity |
| Defect Priority has defined the order in which the developer should resolve a defect | Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect must be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect must be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |
| Priority is categorized into three types   * Low * Medium * High | Severity is categorized into five types   * Critical * Major * Moderate * Minor * Cosmetic |

**What is Bug Life Cycle?**

* “A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result. Bugs arise from mistakes and errors, made by people, in either a program’s source code or its design.”
* The duration or time span between the first-time defects is found and the time that it is closed successfully, rejected, postponed, or deferred is called as Defect Life Cycle.
* When a bug is discovered, it goes through several states and eventually reaches one of the terminal states, where it becomes inactive and closed.
* The process by which the defect moves through the life cycle is depicted next slide.
* **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.”

**Explain the difference between Functional testing and Nonfunctional testing**

|  |  |
| --- | --- |
| Functional testing | Nonfunctional testing |
| Carried out to validate software actions. | It is done to validate the performance of the software. |
| Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| Functional testing is easy to execute by manual testing. | It is very hard to perform non-functional testing manually. |
| It describes what the product does. | It describes how the product works. |
| Check login functionality. | the dashboard should load in 2 seconds. |
| It is performed before non-functional testing. | It is performed after the functional testing. |
| It is based on customer’s requirements. | It focusses on customer’s expectation. |
| It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| Examples of Functional Testing Types   * Unit testing * Smoke testing * User Acceptance * Integration Testing * Regression testing * Localization * Globalization * Interoperability | Examples of Non-functional Testing Types   * performance Testing * Volume Testing * Scalability * Usability Testing * Load Testing * Stress Testing * Compliance Testing * Portability Testing * Disaster Recover Testing |

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

|  |  |
| --- | --- |
| 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, a greater number of members (developers) are required for the whole process. | In STLC, a smaller number of members (testers) are needed. |
| 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. |
| Creation of reusable software systems is the result of SDLC. | A tested software system is the result of STLC. |

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

* Difference between test scenarios and test cases is that
* **Test Scenarios:**  A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility.
* **Test Cases:**  It is a document that contains the steps that must be executed, it has been planned earlier.
* **Test Script:**It is written in a programming language and it is a short program used to test part of functionality of the software system. In other words, a written set of steps that should be performed manually.

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

* describing the scope, approach, resources, and schedule of intended test activities. document
* Determining the scope and risks, and identifying the objectives of testing.
* Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.
* Integrating and coordinating the testing activities into the software life cycle activities:
* Acquisition, supply, development, operation, and maintenance.
* Making decisions about what to test, what roles will perform the test activities, how the test activities should be done, and how the test results will be evaluated?
* Scheduling test analysis and design activities.
* Scheduling test implementation, execution, and evaluation.
* Assigning resources for the different activities defined
* Defining the amount, level of detail, structure, and templates for the test documentation.

##### Test planning Strategy

All projects require a set of plans and strategies which define how the testing will be conducted.

##### Test planning Factor

* Factors which affect test planning
* The organization’s test policy
* Scope of the testing being performed
* Testing objectives
* Project Risks – e.g., business, technical, people
* Constraints – e.g., business imposed, financial, contractual etc.
* Criticality (e.g., system/component level)
* Testability
* Availability of resources
* Test plans are continuously refined
* As more information becomes available
* As new risks arise or others are mitigated
* Not set in concrete, but changes must be carefully managed

##### Test planning Activity

**Approach:** Defining the overall approach of testing (the test strategy), including the definition of the test levels and entry and exit criteria.

Integrating and coordinating the testing activities into the software life cycle

* Activities: acquisition, supply, development, operation, and maintenance.
* Making decisions about
* What to test
* Who does test? I.e., what roles will perform the test activities
* When and how the test activities should be done and when they should be stopped (exit criteria – see next slides)
* How the test results will be evaluated
* Assigning resources for the different tasks defined.

##### Exit Criteria

* How do we know when to stop testing?
* Run out of time?
* Run out of budget?
* The business tells you it went live last night!
* Boss says stop?
* All defects have been fixed?
* When out exit criteria have been met?
* Purpose of exit criteria is to define when we STOP testing either at the:
* End of all testing – i.e., product Go Live

End of phase of testing (e.g., hand over from System Test to UAT)

**What is priority?**

Priority is Relative and Business-Focused. Priority defines the order in which we should resolve a defect. Should we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer must fix it at the earliest. The priority status is set based on the customer requirements.

For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

**There are four types of priority as shown below:**

**Low:** The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect has been fixed.

**Medium:** The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.

**High:** The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the repair has been done.

**Critical:** Extremely urgent, resolve immediately

**What is severity?**

##### Defect Severity

* Severity is absolute and Customer-Focused. It is the extent to which the defect can affect the software. In other words, it defines the impact that a given defect has on the system.
* For example: If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by a user is rare but the impact of application crashing is severe. So, the severity is high but priority is low.

**Severity can be of following types:**

* + - **Critical:** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
    - **Major (High):** The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
    - **Moderate (Medium**): The defect that does not result in the termination, but causes the system to produce incorrect, incomplete, or inconsistent results then the severity will be stated as moderate.
    - **Minor (Low):** The defect that does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
    - **Cosmetic:** The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

**Bug categories are…**

* A Bug Report in Software Testing is a detailed document about bugs found in the software application. Bug report contains each detail about bugs like description, date when bug was found, name of tester who found it, name of developer who fixed it, etc. Bug report helps to identify similar bugs in future so it can be avoided.
* While reporting the bug to developer, your Bug Report should contain the following information
* **Defect ID** – Unique identification number for the defect.
* **Defect Description** – Detailed description of the Defect including information about the module in which Defect was found.
* **Version** – Version of the application in which defect was found.
* **Steps** – Detailed steps along with screenshots with which the developer can reproduce the defects.
* **Date Raised** – Date when the defect is raised
* **Reference**– where in you Provide reference to the documents like. requirements, design, architecture or maybe even screenshots of the error to help understand the defect
* **Detected By** – Name/ID of the tester who raised the defect
* **Status** – Status of the defect, more on this later
* **Fixed by** – Name/ID of the developer who fixed it
* **Date Closed** – Date when the defect is closed
* **Severity** which describes the impact of the defect on the application
* **Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

**Advantage of Bugzilla.**

There are several advantages in Bugzilla:

* Advanced search capabilities
* E-mail Notifications
* Modify/file Bugs by e-mail
* Time tracking
* Strong security
* Customization Localization

**Difference between priority and severity**

|  |  |
| --- | --- |
| Priority | Severity |
| Defect Priority has defined the order in which the developer should resolve a defect. | Defect Severity is defined as the degree of impact that a defect has on the operation of the product. |
| Priority is associated with scheduling. | Severity is associated with functionality or standards. |
| Priority indicates how soon the bug should be fixed. | Severity indicates the seriousness of the defect on the product functionality. |
| Priority of defects is decided in consultation with the manager/client. | QA engineer determines the severity level of the defect. |
| Priority is driven by business value. | Severity is driven by functionality. |
| Its value is subjective and can change over a period depending on the change in the project situation. | Its value is objective and less likely to change. |
| High priority and low severity status indicates, defect must be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect must be fixed but not on immediate bases |
| Priority status is based on customer requirements. | Severity status is based on the technical aspect of the product. |
| During UAT the development team fix defects based on priority. | During SIT, the development team will fix defects based on the severity and then priority. |
| Priority is categorized into three types   * Low * Medium * High | Severity is categorized into five types   * Critical * Major * Moderate * Minor * Cosmetic |

**What are the different Methodologies in Agile Development Model?**

There are several methodologies in agile development model:

* **Individuals and interactions, Over processes and tools**

Suppose the team finds any issue in software then they search for another process or tool to resolve the issue. But, in Agile, it is preferable to interact with client, manager or team regarding issue and make sure that the issue gets resolved.

* **Working software, over comprehensive documentation**

Documentation is needed, but working software is much needed. Agile is not saying that documentation is not needed, but working software is much needed. For example, you have 20page documents, but you do not have a single prototype of the software.

* **Customer collaboration, Over contract negotiation**

Contract negotiation is important as they make the budget of software, but customer collaboration is more important than over contract negotiation. For example, if you stuck with the requirements or process, then do not go for a contract which we have negotiated. You need to interact with the customer, gather their requirements.

* **Responding to change, over following a plan**

In the waterfall model, everything is planned, i.e., at what time, each phase will be completed. Sometimes you need to implement the new requirements in the middle of the software, so you need to be versatile to make changes in the software.

**Write a Scenario of Pen**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | Verify the type of pen, whether it is a ballpoint pen, ink pen, or gel pen. |
| 2 | Verify that the user can write clearly over different types of papers. |
| 3 | Check the weight of the pen. It should be as per the specifications. In case not mentioned in the specifications, the weight should not be too heavy to impact its smooth operation. |
| 4 | Verify if the pen is with a cap or without a cap. |
| 5 | Verify the color of the ink on the pen. |
| 6 | Check the odor of the pen’s ink on writing over a surface. |
| 7 | Verify the surfaces over which the pen is able to write smoothly apart from paper e.g., cardboard, rubber surface, etc. |
| 8 | Verify that the text written by the pen should have consistent ink flow without leaving any blob. |
| 9 | Check that the pen’s ink should not leak in case it is tilted upside down. |
| 10 | Verify if the pen’s ink should not leak at higher altitudes. |
| 11 | Verify if the text written by the pen is erasable or not. |
| 12 | Check the functioning of the pen by applying normal pressure during writing. |
| 13 | Verify the strength of the pen’s outer body. It should not be easily breakable. |
| 14 | Verify that text written by pen should not get faded before a certain time as mentioned in the specification. |
| 15 | Check if the text written by the pen is waterproof or not. |
| 16 | Verify that the user is able to write normally by tilting the pen at a certain angle instead of keeping it straight while writing. |
| 17 | Check the grip of the pen, and whether it provides adequate friction for the user to comfortably grip the pen. |
| 18 | Verify if the pen can support multiple refills or not. |
| 19 | For ink pens, verify that the mechanism to refill the pen is easy to operate. |
| 20 | In the case of a ballpoint pen, verify the size of the tip. |

**Write a scenario of only WhatsApp chat messages**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | Verify your internet connection is on |
| 2 | Verify your device is android |
| 3 | Verify you have valid and active WhatsApp account |
| 4 | Verify open WhatsApp application on your device |
| 5 | Verify login to your account using your mobile number and verification code |
| 6 | Verify tap on the chat icon at the bottom right corner of the screen to open the chat window |
| 7 | Verify that the chat window opens properly and all existing conversation are displayed correctly |
| 8 | Verify start a new chat with a contact in your list by tapping the new chat button |
| 9 | Verify type a message in the chat box and send it |
| 10 | Verify that the message is send successfully and the delivery status is displayed as send |
| 11 | Verify wait for the recipient to receive the message and check if the message is delivered successfully |
| 12 | Verify check the time and date stamp of the message to ensure that it is correct |
| 13 | Verify send a voice message and verify that it is recorded and send successfully |
| 14 | Verify send an image or video file and verify that it is send successfully and can be viewed by the recipient |
| 15 | Verify that the chat interface is easy to use and navigate |
| 16 | Verify test the search functionality of the chat by searching for a specific message or conversation |
| 17 | Verify test the chat backup and restore functionally by creating a backup and then restoring it |
| 18 | Verify test the group chat functionality by creating a new group and adding contacts to it |
| 19 | Verify test the mute and notification settings for individual chats and groups |
| 20 | Verify that the app is responsive and does not crash during the test |

**Write a Scenario of Pen Stand**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | verify the material of pen stand like plastic, wood, metal, etc. |
| 2 | verify the size of the pen stand |
| 3 | verify the shape of the pen stand |
| 4 | verify that how many pen can we put in it |
| 5 | verify the color of the pen stand |
| 6 | verify the no. of the compartments in the pen stand |
| 7 | verify the design of the pen stand |
| 8 | verify the no. of small compartments for other tiny objects like eraser, sharpener, cello tape, paper clips, etc. |
| 9 | verify that pen stand has clock or not |
| 10 | verify that pen stand has calendar or not |
| 11 | verify that pen stand has proper height |
| 12 | verify the alignment of the legs of pen stand |
| 13 | verify the has card holder or not |
| 14 | verify the weight of the pen stand |

**Write a Scenario of Door**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | Verify the type of door weather it is single door double door etc. |
| 2 | Verify that the door is made up of wooden metal etc. |
| 3 | verify the door is opening inward or outward. |
| 4 | Verify that the dimension of the doors is as per the specifications |
| 5 | Verify the colour of the door is as per requirement or not. |
| 6 | verify the door is a leading door washroom door any other location door |
| 7 | verify if the door is built with glass, then it is a transparent or non-transparent door |
| 8 | verify that company logo is displayed on the door lock or not. |
| 9 | verify whether the door makes any sounds while opening or closing |
| 10 | Verify that the stopper is available on the door not. |
| 11 | Verify that the door is waterproof or not |
| 12 | verify that the door can be opened by children with less force or not |
| 13 | check that automatic door closure device is working or not |
| 14 | verify the length width and height of the door is as per requirement. |
| 15 | verify that the door has not any cut so that any child may not harm by it. |
| 16 | verify the door is a leading door washroom door any other location door |
| 17 | verify the type of locks in the door |
| 18 | verify that peek hole is present or not. |
| 19 | Verify that the door is washable or easily cleanable. |
| 20 | Verify that cloth hanger is present or not. |

**Write a Scenario of ATM**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | verify the type of the ATM machine whether touch screen or manual button ATM machine |
| 2 | verify if valid card inserts ATM machine, screen should ask you to insert pin |
| 3 | verify that case dispenser is working as expected |
| 4 | verify if the receipt printer is working properly and receipt come out properly. |
| 5 | verify if the screen buttons are working correctly |
| 6 | verify if optional touch screen is working as per expectation |
| 7 | verify if all text button on screen is visible clearly |
| 8 | verify every number button on the keypad |
| 9 | verify the cancel button on the keypad |
| 10 | verify keypad number button colour should be visible clearly |
| 11 | verify on screen text colour and font, user should be read clearly |
| 12 | verify the language selection option if the data is display in selected language |
| 13 | verify insert card, correct pin, and print the receipt for available balance |
| 14 | verify after valid transaction whether the printed data is correct or not |
| 15 | verify time to system log out time |
| 16 | verify using different bank card |
| 17 | verify the massage after enter correct card and incorrect pin |
| 18 | verify the message there is no case in the ATM |
| 19 | verify message after transaction |
| 20 | verify log out after finished user transaction |

**When to used Usability Testing?**

Usability Testing identifies usability errors in the development cycle and can save a product from failure system early in

Need For Usability Testing

Aesthetics and design are important.

Howell a product looks usually determines how well it works.

There are many software applications / websites, which miserably fail, once launched, due to following reasons –hereto I click next?

Which page needs to be navigated?

Which Icon or Jargon represents what?

Error messages are not consistent or effectively displayed Session time not sufficient.

**What is the procedure for GUI Testing?**

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

**Write a scenario of Microwave Owen**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 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 oven heats food at the desired temperature within a specified time duration. |
| 5 | Verify the ovens functioning with maximum attainable temperature. |
| 6 | Verify the ovens functioning with minimum attainable temperature. |
| 7 | Verify that the oven’s plate rotation is 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 oven’s functionality with different kinds of food – solid, 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. |

**Write a scenario of Coffee vending Machine**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | verify the turn on the coffee vending machine and wait for it to finish its startup process |
| 2 | verify that the dimension of the coffee machine is as per the specification |
| 3 | Verify that outer body of coffee vending machine |
| 4 | verify the inner part’s material, is as per the specification |
| 5 | verify the machine’s body colour is correctly visible and as per specification |
| 6 | verify the brand is correctly visible and as per specification |
| 7 | verify the input mechanism for coffee ingredients-milk, water, coffee beans/powder, etc |
| 8 | Verify that the quantity of hot water, milk, coffee powder per serving is correct |
| 9 | Verify the power/voltage requirements of the machine |
| 10 | Verify the effect of suddenly switching off the machine or cutting the power |
| 11 | verify the switching off or power cutting situation remaining coffee should not get come out of the nozzle. |
| 12 | Verify that coffee should not leak when not in operation |
| 13 | Verify the amount of coffee served in single-serving is as per specification |
| 14 | Verify that the digital display displays correct information |
| 15 | Verify the indicator lights when the machine is switched on-off |
| 16 | Verify that the functioning of all the buttons work properly when pressed |
| 17 | Verify that each button has an image/text with it, indicating the task it performs |
| 18 | Verify that complete quantity of coffee should get poured in a single operation |
| 19 | verify the no residual coffee should be present in the nozzle |
| 20 | Verify the mechanism to clean the system work correctly- former |
| 21 | Verify that the coffee served has the same and correct temperature each time it is served by the machine |
| 22 | verify that insert a bill of a specified denomination into the bill acceptor and ensure that it is accepted |
| 23 | verify select the type of coffee you want to order from the machine's touch screen interface |
| 24 | verify the choose any additional options such as sugar or cream |
| 25 | verify the confirm your order and wait for the coffee to be dispensed |
| 26 | verify taste the coffee to ensure that it is of the desired quality and consistency |
| 27 | verify the above steps with different types of coffee to ensure that the vending machine can dispense different varieties correctly |
| 28 | verify test the machine's ability to handle multiple orders |
| 29 | verify the machine's ability to handle any errors, such as a jammed |

**Write a scenario of chair**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | verify the type of chair whether it is simple chair, folding chair, wheel chair, deck chair etc… |
| 2 | verify the material of the chair whether it is wooden, steel, plastic, bag beans etc… |
| 3 | verify the chair leg are level to the floor |
| 4 | verify the chair is stable to take an average human load |
| 5 | verify if there is back support in the chair |
| 6 | verify if there is support for hands in the chair |
| 7 | verify the chair colour and types of paint |
| 8 | verify washing chair with water and check effect on chair |
| 9 | verify the chair weight is normal for easy to moving |
| 10 | verify the chair material is breakable or not |
| 11 | verify the chair dimension is suitable or not |
| 12 | verify the working of chair wheel |
| 13 | verify the chair shape is comfort for eating |
| 14 | verify the temperature effect on chair |
| 15 | verify chair armrest |
| 16 | verify the chair space of sitting |
| 17 | verify the chair colour is as per requirement |
| 18 | verify the chair hydraulic working properly |
| 19 | verify the chair back spring working properly |

**Write a Scenario of Wrist Watch**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | Verify the type of watch analog or digital |
| 2 | Verify in case of analog watch, correctness time displayed by the second, minute and hour hand of the watch |
| 3 | Verify in case of digital watch, digital display for hours, minutes and seconds in correctly displayed |
| 4 | Verify the material of the watch and its strap |
| 5 | Verify 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 | Verify the watch is waterproof |
| 9 | Verify that the numbers in the dial are clearly visible |
| 10 | Verify the watch is having a date and day display |
| 11 | Verify the colour of the displayed in the watch – time, day, date, and other information |
| 12 | Verify that clock’s time can be corrected using key in case an analog clock and buttons in case of a digital clock  Verify the second hand of the watch makes ticking sound |
| 13 | Verify the brand of the watch and its visible in the dial |
| 14 | Verify the clock is having stopwatch, times, and alarm functionality |
| 15 | Verify in case of a digital watch the format of the watch 12 hours or 24 hours |
| 16 | Verify the watch comes with any guarantee or warranty |
| 17 | Verify the dial has glass covering or plastic and the material is breakable |
| 18 | Verify the dial’s glass/plastic is resistant to minor scratches |
| 19 | Verify the battery requirement of the watch |

**Write a Scenario of Lift (Elevator)**

|  |  |
| --- | --- |
| TEST SCENERIO ID | TEST SCENERIO (POSITIVE) |
| 1 | Verify the dimensions of the lift |
| 2 | Verify the type of door of the lift is as per the specification |
| 3 | Verify the type of metal used in the lift interior and exterior |
| 4 | Verify the capacity of the lift in terms of the total weight |
| 5 | Verify the buttons in the lift to close and open the door and numbers as per the number of floors |
| 6 | Verify that lift moves to the floor as the button of the floor is clicked |
| 7 | Verify that lift stops when up/down buttons at floor are pressed |
| 8 | Verify if there is an emergency button to contact officials in case of any mishap |
| 9 | Verify the performance of the floor – the time is taken to go to a floor |
| 10 | Verify that in case of power failure, lift does not free-fall and get halted in the floor |
| 11 | Verify lifts working in case button to open the door is pressed before reaching the destination floor |
| 12 | Verify that inside lift user is prompted with current floor and direction information the lift is moving towards- audio/visual prompt |
| 13 | 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 |
| 14 | Verify the time duration for which door remain open by default |
| 15 | Verify if lift interior is having proper air ventilation |
| 16 | Verify lighting in the lift |
| 17 | Verify that at no point lifts door should open while in motion |
| 18 | Verify that in case of power loss, there should be a backup mechanism to safely get into a floor or a backup power supply |
| 19 | Verify that in case multiple floor number button is clicked, lift should stop at each floor |
| 20 | Verify that in case of capacity limit is reached users are prompted with warning alert- audio/visual |

**Write a Scenario of Gmail-receiving mail (negative & positive)**

|  |  |  |
| --- | --- | --- |
| TEST SCENARIO ID | TEST SCENARIO (POSITIVE) | TEST SCENARIO (NEGATIVE) |
| 1 | Verify that a newly received email is displayed as highlighted in the inbox section | Verify that user has no internet connection |
| 2 | Verify that a newly received email has correctly displayed sender email id or name, mail subject and mail body | Verify that internet connection is not working condition |
| 3 | Verify that on clicking the newly received email, the user is navigated to email content | Verify that user has entered incorrect login credentials |
| 4 | Verify that the email contents are correctly displayed with the desire source formatting | Verify that user has entered incorrect password credentials |
| 5 | Verify that any attachments are attached to the email and are downloadable | Verify that if Gmail fails to receive an email |
| 6 | Verify that the attachments are scanned for viruses before download | Verify that the sender has not send the email |
| 7 | Verify that all the emails marked as read are not highlighted | Verify that the sender has not composed and send the email to the recipient’s email address |
| 8 | Verify that all the emails read as well as unread have a mail read time appended at the end on the email list displayed in the inbox section | Verify the email has been marked as spam |
| 9 | Verify that count of unread emails is displayed alongside inbox text in the left sidebar of Gmail | Verify Gmail has a robust spam filter that automatically marks suspicious emails as spam |
| 10 | Verify that unread email count increase by one on receiving a new email | Verify Gmail server issues, sometimes gmail’s server may experience technical difficulties leading to the failure to receive email |
| 11 | Verify that unread email count decrease by one on reading an email | Verify if the email attachment or the entire email exceeds the maximum size limit set by Gmail, the recipient may not receive the email |
| 12 | Verify that email recipients in cc are visible to all users | Verify that the incorrect email address |
| 13 | Verify that email recipients in bcc are not visible to the user | Verify if the sender has entered an incorrect email address the email will not be delivered to the intended received |
| 14 | Verify that all received emails get piled up in the inbox section and get deleted in cyclic fashion based on the size availability | Verify if the recipient has set up email forwarding, the email may not be received in the Gmail inbox |
| 15 | Verify that email can be received from Gmail email ids like yahoo and Hotmail etc. | Verify that the Gmail filter the social mails |

## **Write a Scenario of WhatsApp group (generate group)**

|  |  |
| --- | --- |
| TEST SCENARIO ID | TEST SCENARIO (POSITIVE) |
| 1 | Verify that open the WhatsApp application on your mobile device |
| 2 | Verify the click on new group button located in the upper right corner of the screen |
| 3 | Verify add profile picture for the group |
| 4 | Verify select the members you want to add to the group from your contact list |
| 5 | Verify the entering group name |
| 6 | Verify once you have added all the desired members, click on the create button to create the new group |
| 7 | Verify test that new group has been created by checking that it appears in your list of WhatsApp groups |
| 8 | Verify test that you can send message to the group by typing a message in the group chat and checking that it delivered to all members of the group |
| 9 | Verify test that you can receive message from the group by asking one of the other group members to send a message and checking that you receive |
| 10 | Verify try to adding members from the group |
| 11 | Verify try to remove members from the group |
| 12 | Verify test the mute and notification setting for the group |
| 13 | Verify experiment with different group setting such as changing group icon |
| 14 | Verify the emoji icon for group name |
| 15 | Verify that the done icon |
| 16 | Verify the disappearing message option |
| 17 | Verify the profile info of the group |
| 18 | Verify the video call button |
| 19 | Verify the voice call button |
| 20 | Verify the emoji buttons on the text box |
| 21 | Verify the WhatsApp pay icon |
| 22 | Verify the camera icon |
| 23 | Verify the audio recorder button |
| 24 | Verify that sharing files in the group |
| 25 | Verify test you can leave the group by clicking on the group name, then clicking on exit group |

## **Write a scenario of Instagram (video call with chat)**

|  |  |
| --- | --- |
| TEST SCENARIO ID | TEST SCENARIO (POSITIVE) |
| 1 | Verify open the Instagram application on phone |
| 2 | Verify that direct message icon on home page |
| 3 | Verify that select a contact with whom you want to video call or chat |
| 4 | Verify initiate a video call by tapping on the video camera icon |
| 5 | Verify that the call is connected |
| 6 | Verify that the video quality |
| 7 | Verify that the audio quality |
| 8 | Verify that during the call try sending text message |
| 9 | Verify that during the call try sending emojis and stickers |
| 10 | Verify that during the call try sending photos through the chat window |
| 11 | Verify make sure the messages are delivered instantly and there is no delays |
| 12 | Verify end the call and check the chat history saved automatically |
| 13 | Verify that restart the app and check the call history |
| 14 | Verify that restart the app and check the chat messages are still available |
| 15 | Verify that start chat conversation and test the sending and receiving of messages |
| 16 | Verify that message is delivered in timely manner |
| 17 | Verify that notifications are accurate and reliable |
| 18 | Verify that the camera button |
| 19 | Verify the audio recorder button |
| 20 | Verify the gallery icon |
| 21 | Verify the stickers icon |

## **To create HLR and test case of only first page of Instagram**

<https://docs.google.com/spreadsheets/d/1mnCaJyNAb337sy10cGjqJOj9jqcI4MB_/edit?usp=share_link&> ouid=109322968411295003916&rtpof=true&sd=true

## **To create HLR and test case of only WhatsApp Web page of WhatsApp**

[**https://docs.google.com/spreadsheets/d/1mnCaJyNAb337sy10cGjqJOj9jqcI4M**](https://docs.google.com/spreadsheets/d/1mnCaJyNAb337sy10cGjqJOj9jqcI4M) **B\_/edit?usp=share\_link&ouid=109322968411295003916&rtpof=true&sd=true**