Module–2(Manual Testing)

• What is Exploratory Testing?

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

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

• What is traceability matrix?

A traceability matrix is a document that details [the technical requirements](https://www.wrike.com/project-management-guide/faq/what-are-technical-requirements-in-project-management/) for a given [test scenario](https://learn.marsdd.com/article/product-testing-methods/#:~:text=Test%20cases%20probe%20each%20way,changes%20that%20might%20be%20necessitated.) and its current state. It helps the testing team understand the level of testing that is done for a given product.

The traceability process itself is used to review the test cases that were defined for any requirement. It helps users identify which [requirements](https://www.wrike.com/blog/requirements-gathering-guide/#What-is-requirements-gathering-in-project-management) produced the most number of defects during a testing cycle.

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

• What is Boundary value testing?

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

In software development, partitioning involves dividing larger code or projects into smaller components to make them more manageable. Without the partitions, the code may have many errors and take a longer time to compile properly. Equivalence partitioning is a software technique that occurs directly before boundary testing and divides input data into equal partitions which function to obtain test cases or the individual test scenarios that developers perform on a system to determine if it functions correctly. The technique is important for boundary testing because it helps reduce the amount of time necessary to test further.

• What is Equivalence partitioning testing?

Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as ECP. It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived.

An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

• What is Integration testing?

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

## Integration Strategies:

* Big-Bang Integration
* Top Down Integration
* Bottom Up Integration
* Hybrid Integration

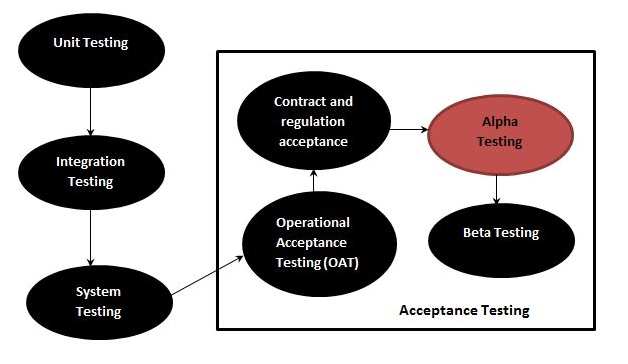
• What determines the level of risk?

The likelihood of an adverse event and the impact of the event.

• What is Alpha testing?

Alpha testing takes place at the developer's site by the internal teams, before release to external customers. This testing is performed without the involvement of the development teams.

The following diagram explains the fitment of Alpha testing in the software development life cycle.



In the first phase of alpha testing, the software is tested by in-house developers during which the goal is to catch bugs quickly.

In the second phase of alpha testing, the software is given to the software QA team for additional testing.

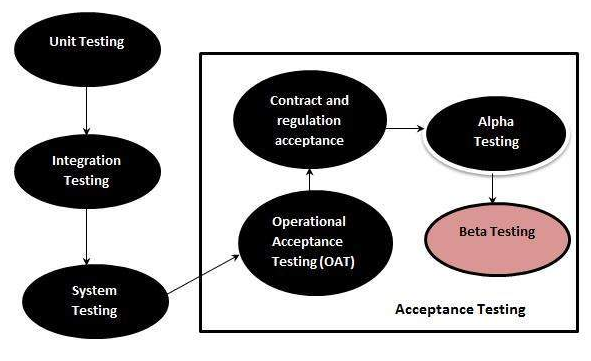
Alpha testing is often performed for Commercial off-the-shelf software (COTS) as a form of internal acceptance testing, before the beta testing is performed.

• What is beta testing?

Beta testing also known as user testing takes place at the end users site by the end users to validate the usability, functionality, compatibility, and reliability testing.

Beta testing adds value to the software development life cycle as it allows the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. These inputs are not only critical to the success of the product but also an investment into future products when the gathered data is managed effectively.

The following diagram explains the fitment of Beta testing in the software development life cycle:



## Beta Testing Dependencies

There are number of factors that depends on the success of beta testing:

* Test Cost
* Number of Test Participants
* Shipping
* Duration of Test
* Demographic coverage

• 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’s also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.

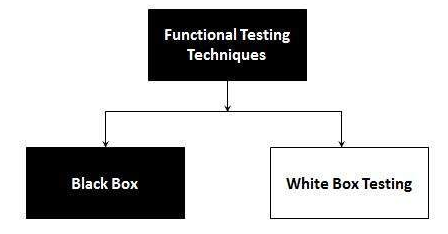
It’s one of most frequent black box testing types which is performed by QA Team.

• What is functional system testing?

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

## **Functional Testing Techniques:**

There are two major Functional Testing techniques as shown below:



The other major Functional Testing techniques include:

* Unit Testing
* Integration Testing
* Smoke Testing
* User Acceptance Testing
* Localization Testing
* Interface Testing
* Usability Testing
* System Testing
* Regression Testing
* Globalization Testing

• What is Non-Functional Testing?

Non-Functional testing is a software testing technique that verifies the attributes of the system such as memory leaks, performance or robustness of the system. Non-Functional testing is performed at all test levels.

## Non-Functional Testing Techniques:

* Baseline testing
* Compatibility testing
* Compliance testing
* Endurance testing
* Load testing
* Localization testing
* Internationalization testing
* Performance testing
* Recovery testing
* Resilience testing
* Security testing
* Scalability testing
* Stress testing
* Usability testing
* Volume testing

• What is GUI Testing?

GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behaviour.

GUI Testing includes the application behaviour towards keyboard and mouse movements and how different GUI objects such as toolbars, buttons, menu bars, dialog boxes, edit fields, lists, behaviour to the user input.

## GUI Testing Guidelines

* Check Screen Validations
* Verify All Navigations
* Check usability Conditions
* Verify Data Integrity
* Verify the object states
* Verify the date Field and Numeric Field Formats
* Following are some of the open source GUI automation tools in the market:

|  |  |  |
| --- | --- | --- |
| **Product** | **Licensed Under** | **URL** |
| AutoHotkey | GPL | http://www.autohotkey.com/ |
| Selenium | Apache | http://docs.seleniumhq.org/ |
| Sikuli | MIT | http://sikuli.org |
| Robot Framework | Apache | www.robotframework.org |
| watir | BSD | http://www.watir.com/ |
| Dojo Toolkit | BSD | http://dojotoolkit.org/ |

* Following are some of the Commercial GUI automation tools in the market.

|  |  |  |
| --- | --- | --- |
| **Product** | **Vendor** | **URL** |
| AutoIT | AutoIT | http://www.autoitscript.com/site/autoit/ |
| EggPlant | TestPlant | www.testplant.com |
| QTP | Hp | http://www8.hp.com/us/en/software-solutions/ |
| Rational Functional Tester | IBM | http://www-03.ibm.com/software/products/us/en/functional |
| Infragistics | Infragistics | www.infragistics.com |
| iMacros | iOpus | http://www.iopus.com/iMacros/ |
| CodedUI | Microsoft | http://www.microsoft.com/visualstudio/ |
| Sikuli | Micro Focus International | http://www.microfocus.com/ |

• 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.

• What is load testing?

Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions.

## Load Testing Approach:

* Evaluate performance acceptance criteria
* Identify critical scenarios
* Design workload Model
* Identify the target load levels
* Design the tests
* Execute Tests
* Analyze the Results

## Objectives of Load Testing:

* Response time
* Throughput
* Resource utilization
* Maximum user load
* Business-related metrics

• What is stress Testing?

Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress.

The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

## Reasons for conducting Stress Testing:

* It allows the test team to monitor system performance during failures.
* To verify if the system has saved the data before crashing or NOT.
* To verify if the system prints meaning error messages while crashing or did it print some random exceptions.
* To verify if unexpected failures do not cause security issues.

## Stress Testing - Scenarios:

* Monitor the system behaviour when maximum number of users logged in at the same time.
* All users performing the critical operations at the same time.
* All users Accessing the same file at the same time.
* Hardware issues such as database server down or some of the servers in a server park crashed.

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

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

## White Box Testing Techniques:

* Statement Coverage - This technique is aimed at exercising all programming statements with minimal tests.
* **Branch Coverage -**This technique is running a series of tests to ensure that all branches are tested at least once.
* **Path Coverage -**This technique corresponds to testing all possible paths which means that each statement and branch is covered.

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

Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle.

This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

There are different techniques involved in Black Box testing.

* Equivalence Class
* Boundary Value Analysis
* Domain Tests
* Orthogonal Arrays
* Decision Tables
* State Models
* Exploratory Testing
* All-pairs testing

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

A software bug arises when the expected result doesn’t 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.

Following are the methods for preventing programmers from introducing bugs during development:

* Programming Techniques adopted
* Software Development methodologies
* Peer Review
* Code Analysis

## Common Types of Defects

Following are the common types of defects that occur during development:

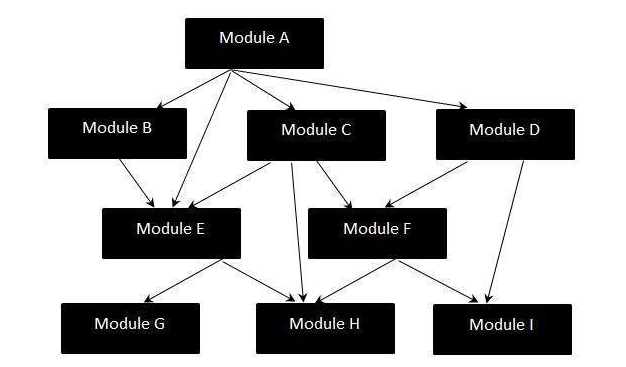
* Arithmetic Defects
* Logical Defects
* Syntax Defects
* Multithreading Defects
* Interface Defects
* Performance Defects

• **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.

## **Big Bang Integration – Work Flow Diagram**

Big Bang Testing is represented by the following workflow diagram:



## **Disadvantages of Big-Bang Testing**

* Defects present at the interfaces of components are identified at very late stage as all components are integrated in one shot.
* It is very difficult to isolate the defects found.
* There is high probability of missing some critical defects, which might pop up in the production environment.
* It is very difficult to cover all the cases for integration testing without missing even a single scenario.

• **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.

## Examples of Exit Criteria:

* 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.

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

Regression testing is a black box testing techniques. It is used to authenticate a code change in the software does not impact the existing functionality of the product. Regression testing is making sure that the product works fine with new functionality, [bug](https://www.javatpoint.com/bug-in-software-testing) fixes, or any change in the existing feature.

Regression testing is a type of [software testing](https://www.javatpoint.com/software-testing-tutorial). Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.

Regression testing can be performed on a new build when there is a significant change in the original functionality. It ensures that the code still works even when the changes are occurring. Regression means Re-test those parts of the application, which are unchanged.

Regression tests are also known as the Verification Method. Test cases are often automated. [Test cases](https://www.javatpoint.com/test-case) are required to execute many times and running the same test case again and again manually, is time-consuming and tedious too.

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

There are **seven fundamental testing principles** which help the testing teams to utilize their time and effort and make the testing process an effective one.

* [Testing shows presence of defects](https://www.edureka.co/blog/seven-principles-of-software-testing/#testingdefects)
* [Exhaustive Testing is not possible](https://www.edureka.co/blog/seven-principles-of-software-testing/#exhaustivetesting)
* [Early Testing](https://www.edureka.co/blog/seven-principles-of-software-testing/#earlytesting)
* [Defect Clustering](https://www.edureka.co/blog/seven-principles-of-software-testing/#defectclustering)
* [Pesticide Paradox](https://www.edureka.co/blog/seven-principles-of-software-testing/#pesticideparadox)
* [Testing is context-dependent](https://www.edureka.co/blog/seven-principles-of-software-testing/#contextdependent)
* [Absence of errors fallacy](https://www.edureka.co/blog/seven-principles-of-software-testing/#errorfallacy)

**1. Testing shows presence of defects**

Testing an application is important to know whether any **error or defect exists**. The objective is to find more and more hidden defects using different techniques and methods. Testing helps you reveal undiscovered defects and if no defects are found then it does not mean that the **software** is **defect-free**.

For example, an application might seem to be error-free after going through different stages of testing. But during the production in the environment, the user may come across any defect which did not occur during the testing.

### ****2. Exhaustive testing is not possible****

It is difficult to test all the functionalities with valid and invalid combinations of input data during actual testing. Therefore, testing of a few combinations is done based on priority using different techniques.**Exhaustive testing** will take **unlimited efforts** and most of those efforts are ineffective. Also, the project timelines would not allow testing of so many combinations.

For example, If you have an input field which accepts alphabets, special characters, and numbers from 0 to 1000 only, it would have numerous combinations for testing. It is not possible to test all combinations for each input type.

### ****3. Early testing****

Testers get involved in the early stage to find defects during the**requirement analysis phase**. The cost involved in fixing such defects is very less when compared to those that are found during the later stages of testing.

Testing needs to be performed on requirement documents, specification or any other type of document so that if requirements are incorrectly defined then it can be**fixed immediately** rather than fixing them in the **development phase**.

### ****4. Defect clustering****

During testing, it is observed that most of the **defects** found are related to a small number of **modules**. There might be multiple reasons for this like the modules may be complex, coding related to such modules may be complicated, etc.

This is the Pareto Principle of software testing where you can detect 80% of the problems in 20% of the modules.

### ****5. Pesticide paradox****

Pesticide Paradox principle states that if you execute the same set of test cases again and again over the period of time then these set of tests cannot identify new defects in the system.

In order to overcome this **Pesticide Paradox**, you need to review and revise the set of test cases regularly. If required a new set of test cases can be added and the existing test cases can be deleted if they are not able to find any more defects from the system.

### ****6. Testing is context-dependent****

There are various domains available in the market and each domain has a number of applications. For each domain, their applications have different requirements, functions, different testing purpose, risk, techniques, etc. Different domains are tested differently and thus the testing is purely based on the **context** of the **domain** or application.

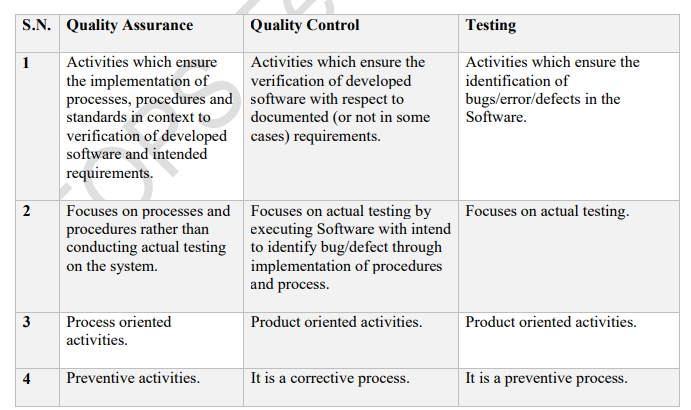
For Example, testing a banking application is different than testing any e-commerce or advertising application. The risk associated with each type of application is different. So, you should not use the same method, technique, and testing type to test different applications.

### ****7. Absence of errors fallacy****

If the software is tested fully and no defects are found before release, then you can consider the software to be 99% defect-free. But if the software is tested against **wrong requirements**, finding defects and fixing them on time would not help as testing is performed on wrong requirements. This would not comply with the user needs.

For Example, suppose you are testing an application related to an e-commerce site and the requirements against “Shopping Cart or Shopping Basket” functionality is wrongly interpreted and tested. Here, finding more defects would not help you in moving the application into the next phase or in the production environment.

• Difference between QA v/s QC v/s Tester



• Difference between Smoke and Sanity?

#### Smoke Testing has a goal to verify “stability” whereas Sanity Testing has a goal to verify “rationality”.

* Smoke Testing is done by both developers or testers whereas Sanity Testing is done by testers.
* Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
* Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
* Smoke testing is documented or scripted whereas Sanity testing isn’t.
* Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component.

• **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. |

• **Explain types of Performance testing.**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

## **Performance Testing Techniques:**

* **Load testing -**It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.
* **Stress testing -**It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
* **Soak testing -**Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
* **Spike testing -**Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

• **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.

Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

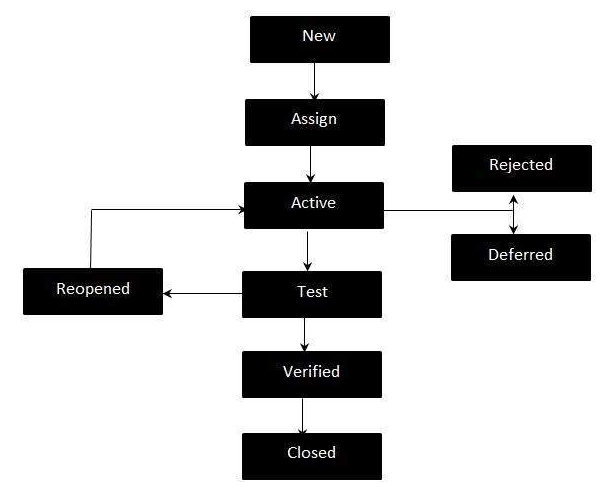
• **Difference between Priority and Severity**

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

• **What is Bug Life Cycle?**

Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.

## Workflow:



## Defect Life Cycle States:

* New - Potential defect that is raised and yet to be validated.
* Assigned - Assigned against a development team to address it but not yet resolved.
* Active - The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.
* Test - The Defect is fixed and ready for testing.
* Verified - The Defect that is retested and the test has been verified by QA.
* Closed - The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.
* Reopened - When the defect is NOT fixed, QA reopens/reactivates the defect.
* Deferred - When a defect cannot be addressed in that particular cycle it is deferred to future release.
* Rejected - A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

• **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 **Examples:**

**2.** Smoke Testing **1.** Performance Testing

**3.** Integration Testing

**4.** Regression Testing

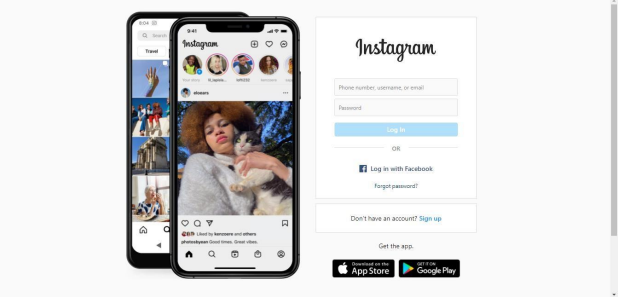
**2.** Load Testing

**3.** Stress Testing

**4.** Scalability Testing

• **To create HLR & TestCase of**

**1)(Instagram , Facebook) only first page**



2) Facebook Login Page : <https://www.facebook.com/>



* **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, 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. |

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

**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 has to be executed, it has been planned earlier.

**Test Script:**It is written in a programming language and it's 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?**

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project.

## **Test Planning Activities:**

* To determine the scope and the risks that need to be tested and that are NOT to be tested.
* Documenting Test Strategy.
* Making sure that the testing activities have been included.
* Deciding Entry and Exit criteria.
* Evaluating the test estimate.
* Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
* The Test artefacts delivered as part of test execution.
* Defining the management information, including the metrics required and defect resolution and risk issues.
* Ensuring that the test documentation generates repeatable test assets.
* **What is priority?**

Priority is defined as the order in which the defects should be resolved. The priority status is usually set by the testing team while raising the defect against the dev team mentioning the timeframe to fix the defect. The Priority status is set based on end users requirement.

For example: If the company logo is incorrectly placed in the company's web page then the priority is high but it is of low severity.

## Priority List:

Priority can be marked as either of the following states:

* **Low -**This defect can be fixed after the critical ones are fixed.
* **Medium -**The defect should be resolved in the subsequent builds.
* **High -**The defect must be resolved immediately because the defect is affecting the application to a considerable extent and the relevant modules cannot be used until it's fixed.
* **Urgent -**The defect must be resolved immediately because the defect is affecting the application or the product severely and the product cannot be used until it has been fixed.

• **What is severity?**

Severity means the seriousness of the defect in the product functionality. The test engineer determines the severity level of the defect. It is driven by functionality.

• **Bug categories are…**

1. **Software Bugs by Nature:**  
   Software bugs have different natures where they affect the overall functioning of the software differently. Though there are dozens of such bugs existing currently, you may not face them frequently. With that in mind, here are the most common software bugs categorized by nature that you are most likely to witness at some point in your software development career.
   * **Performance Bugs:**  
     No user wants to use software with poor performance. Software bugs that lead to degraded speed, stability, increased response time, and higher resource consumption are considered performance bugs. The most significant sign of any such bug in software is by noticing slower loading speed than usual or analyzing the response time. If any such sign is found, the developer may begin diagnosing a performance bug. The [performance testing](https://www.thinksys.com/performance-testing-services/) phase is part of the development process where every such bug is detected in the software.
   * **Security Bugs:**  
     While using software, security is the biggest concern of a user. Software with poor security will not only put the user’s data at risk but will also damage the overall image of the organization which may take years to recuperate. Due to their high severity, security bugs are considered among the most sensitive bugs of all types. Though it is self-explanatory, security bugs may make the software vulnerable to potential cyber threats. Sometimes, the software organization may not notice such attacks whereas in some cases, these attacks could cause monetary loss to the users, especially small and medium-scale businesses. XSS vulnerabilities, logical errors, and encryption errors are some of the commonest security bugs found in the software. Developers put special focus on checking the code to find any underlying [security bug](https://www.thinksys.com/security-testing-services/) to minimize the risk of cyber-attacks.
   * **Unit Level Bugs:**  
     Unit level bugs are fairly common in software development and do not cause much damage to it as well. Facing basic logic bugs or calculation errors are considered unit-level bugs. The [testing team](https://www.thinksys.com/software-testing-services/) along with the agile team test a small part of the code as a whole. The reason why this testing method is preferred is to make sure that the entire code is working as it is meant to. While testing, the team may encounter unit-level bugs which can be fixed easily as the team is only working with a small code.
   * **Functional Bugs:**  
     Software is as good as the feature it provides. If any of the functionality of a software is compromised, the number of users will start to decline drastically until it becomes functional again. A functional bug is when a certain feature or the entire software is not functioning properly due to an error. The severity of such bugs depends on the feature they are hampering. For instance, an unresponsive clickable button that is not functioning is not as severe as the entire software not working. [Functional testing](https://www.thinksys.com/functional-testing-services/) is done by the testing team to identify any such software bug causing functionality errors. Once identified, the team decides its further classification and severity.
   * **Usability Bugs:**  
     Probably one of the most catastrophic bugs for software, a usability bug or defect can stop the software from working to its potential or make it entirely unusable. Examples of this bug in software testing are the inability to log in to the user account or the inefficient layout of the software for the user. The bottom line is that this type of defect or bug can make it complex for the user to use the software efficiently. The developers and engineers have to look out for the right [usability requirements](https://www.thinksys.com/qa-testing/complete-guide-to-usability-testing/) while testing the code to identify such bugs.
   * **Syntax Errors:**  
     Syntax errors are among the commonest software bug types and do not allow the application to be compiled appropriately. This bug occurs due to an incorrect or missing character from the source code due to which the compiling will be affected. A small error like a missing bracket could lead to this problem. The development or testing team will get to know about this bug during compiling and will further analyze the source code to fix the missing or wrong characters.
   * **Compatibility Errors:**  
     Whenever a software or an application is not compatible with hardware, or an operating system, it is considered as incompatible software or a compatibility error. Finding a compatibility error is not a common practice as they may not show up in the initial testing. Due to this reason, the developers should go for compatibility testing to make sure that their created software is compatible with common hardware and operating systems.
   * **Logic Bugs:**  
     Another one of the most frequently found bugs in a software code, logic errors make the software give wrong output, software crash or failure. In the majority of cases, these bugs are caused due to coding errors where it may make the software stuck in a never-ending loading loop. In that case, only an external interruption or software crashing are the two only things that can break the loading loop.
2. **Priority-Based Software Bugs:**  
   The foremost category here is priority-based software bugs. These are based on the impact these bugs leave on the business. Here, the developers will analyze the bug to determine its impact and its defect priority. Afterward, the timeline is given to each bug where it should be rectified within the stipulated time frame to minimize the bug effect on the user. Here are the four types of priority-based software bugs.
   * **Low-priority defects:**  
     Low priority defects do not cause much impact on the functioning of the application. Rather, they are more about software aesthetics. For instance, any issue with the spelling or the alignment of a button or text could be a low-priority defect. The software testing will move to the exit criteria even if the low-priority defects are not fixed, but they should be rectified before the final release of the software.
   * **Medium-priority defects:**  
     Akin to low-priority defects, medium-priority defects do not cause any significant impact on the software, but they should be fixed in any subsequent or upcoming release. Such defects may not have the same effect for every user and it may vary with the device as well as specific configuration they have.
   * **High-priority defects:**  
     Unlike the previous two, the exit criteria of high-priority defects are not met until the issue is resolved. Every bug falling in this category may make certain features of the software unusable. Even though it may not affect every user, it is mandatory to fix these bugs before any further step is taken in software development or testing.
   * **Urgent Defects:**  
     As the name suggests, all bugs that should be dealt with utmost urgency fall under this category. Urgent defects may leave a lasting impact on the brand image as well as affect the user experience drastically. The stipulated timeline for fixing these bugs is within 24-hours of reporting.
3. **Software Bugs by Severity:**  
   Depending on the technical effect that the bug will cause on the software, the bugs are categorized into four categories.
   * **Low Severity Bugs:**  
     Low severity bugs do not cause much damage to the functioning of the software as their primary target is the user interface. For instance, the font of the text on the program differs from what was used. These bugs can be fixed easily and are nothing to worry about.
   * **Medium Severity Bugs:**  
     Every bug that can affect the functionality of the software a little bit is considered a medium severity bug. All such bugs make the software function different from what it is supposed to function. Though they are not also major for the program, they should be fixed for a better user experience.
   * **High Severity Bugs:**  
     High severity bugs affect the software functionality, making it behave differently from what it was programmed for. Not only are such bugs damaging for the software, they sometimes make the entire software unusable for the user.
   * **Critical Bugs:**  
     Critical bugs are the most damaging bugs in the category that can hinder the functionality of the entire software. The reason why critical bugs are considered the most damaging is that further testing on the software becomes impossible till such bugs exist in the software.

• **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 withtest management instruments;
* it integrates with an e-mailing system;
* it automates documentation.

Not all testers like to work with [Bugzilla](https://qatestlab.com/approach/technologies/bug-tracking-tools/). Some of them find that its interface is too sophisticated, complain that it is hard to manage the logged in errors and that there are many defects in the code of this instrument.

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

Agile refers to the methods and best practices for organizing projects based on the values and principles documented in the [Agile Manifesto](https://www.agilealliance.org/agile101/what-is-agile/). However, there’s no one right way to implement Agile and many different types of methodologies from which to choose. Here are some of the most common Agile frameworks.

#### Kanban

Kanban is a simple, visual means of managing projects that enables teams to see the progress so far and what’s coming up next. Kanban projects are primarily managed through a Kanban board, which segments tasks into three columns: “To Do,” “Doing,” and “Done.”

#### Scrum

Scrum is similar to Kanban in many ways. Scrum typically uses a Scrum board, similar to a Kanban board, and groups tasks into columns based on progress. Unlike Kanban, Scrum focuses on breaking a project down into sprints and only planning and managing one sprint at a time. Scrum also has unique project roles: Scrum master and product owner.

#### Extreme Programming (XP)

Extreme Programming (XP) was designed for Agile software development projects. It focuses on continuous development and customer delivery and uses intervals or sprints, similar to a Scrum methodology. However, XP also has 12 supporting processes specific to the world of software development:

* Planning game
* Small releases
* Customer acceptance tests
* Simple design
* Pair programming
* Test-driven development
* Refactoring
* Continuous integration
* Collective code ownership
* Coding standards
* Metaphor
* Sustainable pace

#### Feature-driven development (FDD)

Feature-driven development is another software-specific Agile framework. This methodology involves creating software models every two weeks and requires a development and design plan for every model feature. It has more rigorous documentation requirements than XP, so it’s better for teams with advanced design and planning abilities. FDD breaks projects down into five basic activities:

* Develop an overall model
* Build a feature list
* Plan by feature
* Design by feature
* Build by feature

#### Dynamic Systems Development Method (DSDM)

The Dynamic Systems Development Method (DSDM) was born of the need for a common industry framework for rapid software delivery. Rework is to be expected, and any development changes that occur must be reversible. Like Scrum, XP, and FDD, DSDM uses sprints. This framework is based on eight fundamental principles:

* Focus on the business need
* Deliver on time
* Collaborate
* Never compromise quality
* Build incrementally from firm foundations
* Develop iteratively
* Communicate continuously and clearly
* Demonstrate control

#### Crystal

Crystal is a family of Agile methodologies that includes Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, etc. Each has a unique framework. Your choice depends on several project factors, such as your team size, priorities, and project criticality.

#### Lean

Lean development is often grouped with Agile, but it’s an entirely different methodology that happens to share many of the same values. The [main principles of the Lean methodology](https://leankit.com/learn/lean/principles-of-lean-development/) include:

* Eliminating waste
* Build quality in
* Create knowledge
* Defer commitment
* Deliver fast
* Respect people
* Optimize the whole

• **Explain the difference between Authorization and Authentication in Web testing.**

| Authentication | Authorization |
| --- | --- |
| In the authentication process, the identity of users are checked for providing the access to the system. | While in authorization process, a the person’s or user’s authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user’s login details. | While it needs the user’s privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines What permission does the user have? |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process. | The OAuth 2.0 protocol governs the overall system of user authorization process. |
| Popular Authentication Techniques-   * Password-Based Authentication * Passwordless Authentication * 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) * Single sign-on (SSO) * Social authentication | Popular  Authorization Techniques-   * Role-Based Access Controls (RBAC) * SON web token (JWT) Authorization * SAML Authorization * OpenID Authorization * OAuth 2.0 Authorization |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. |
| **Example**: Employees in a company are required to authenticate through the network before accessing their company email. | **Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access. |

**What are the common problems faced in Web testing?**

# 5 Web Application Testing Challenges

Testing is a fundamental challenge when developing web applications. Different browsers, interfaces, security threats, and overall app integration are just a few of the issues faced by developers. Since testing is a crucial phase in the development process, the developer should expect to run into unforeseen issues associated with both the web application, and the testing process itself. Each primary issue associated with testing has the potential to branch into uncountable related issues, all of which can be solved if properly identified. Below are five web application testing challenges faced by web developers during the development process.

## Integration

Integration testing exposes problems with interfaces among different program components before deployment. Additionally, integration testing can show the different issues an application may have when interacting with other applications, allowing the developer to tweak things. Environment and infrastructure inconsistency, different interaction models, and overall performance are just a few of the issues associated with integration testing.

## Interoperability

Proving end-to-end functionality between communicating systems is always a challenging obstacle. Different users utilize different browsers and operating systems. To pull data, testing each one to confirm a clear information pathway is very important. Even if the browsers are similar, the web application may be rendered differently based on screen resolution and overall software configuration. This can present some serious issues for developers.

## Security

In one of the most important tests, the developer must make sure that the continually evolving cyber threat can be countered and neutralized. Additionally, tests associated with data integrity before and after an attack are equally important when considering data breaches or lost information. Some of the challenges associated with security testing include dealing with unsecured communications, removing malicious files (if security firewalls have been breached), and the utilization (and integration) of different authentication procedures.

## Performance

Slow applications are not successful. Developers understand that the speed of the app is defined by the need of the user, and with more users expecting more speed, the requirement of performance is non-negotiable. Testing large applications on minimal hardware, underestimating software requirements, and overextending application features are just a few of the issues associated with performance testing. Integration and interoperability issues can also have a direct effect on performance, and because of that, should be tested at the beginning.

## Usability

Since web-based applications are dependent on different browsers, consistent usability is crucial. Additionally, since the app is the brand (or a component thereof), any inconsistency within the user experience may translate into a negative experience, affecting the brand and its potential growth. When testing usability, developers face issues with scalability and interactivity. Since every user is different, it is important for developers to utilize a representative group to test the application across different browsers, using different hardware.

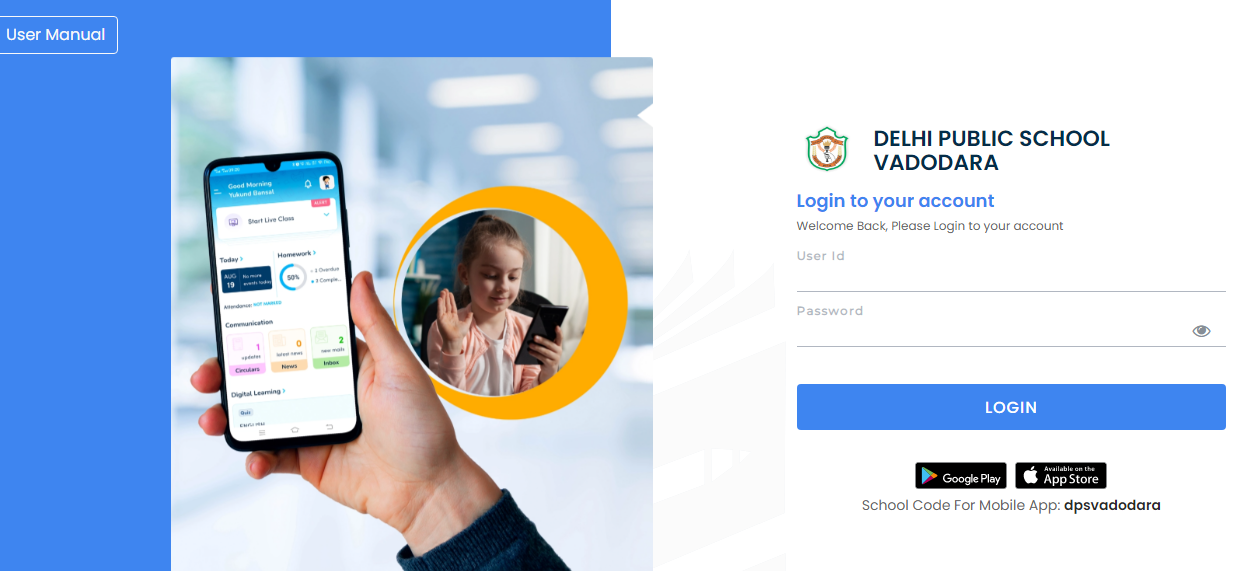
The testing process is inherently full of issues, but each issue faced by the developer is one less that will be faced by the user. In app development, the first interaction with the app needs to be as flawless as possible, allowing the user to adopt and use the app in a way that is as effortless as possible. By testing integration, interoperability, security, performance, and usability, developers will be better able to identify and manage the actual issues associated with the app. In the end, it is about a positive user experience.

**To create HLR & TestCase of WebBased (WhatsApp web , Instagram) 1. WhatsAppWeb :** [**https://web.whatsapp.com**](https://web.whatsapp.com)



/To create HLR and TestCase on this Link.

https://dpsvadodara.edunext2.com/Index



• Write a scenario of only Whatsapp chat messages

• Write a Scenario of Pen

• Write a Scenario of Pen Stand

• Write a Scenario of Door

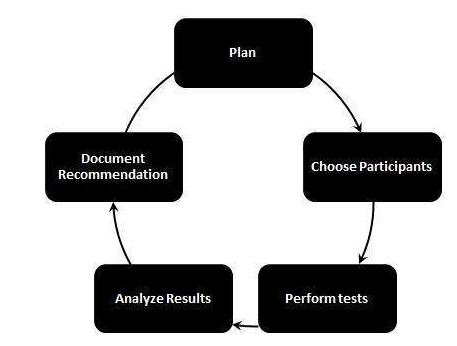
• Write a Scenario of ATM

• **When to used Usablity Testing?**

Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

* Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.
* Time required to get used to in using the software.
* The measure of increase in user productivity if any.
* Assessment of a user's attitude towards using the software.

## **Usability Testing Process:**



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

**Graphical User Interface Testing (GUI) Testing** is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application.GUI testing generally evaluates a design of elements such as layout, colors and also fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links, and content. GUI testing processes may be either manual or automatic and are often performed by third-party companies, rather than developers or end users.

**Feature of Graphical User Interface Testing (GUI):**   
There are some feature of GUI which are given below:

* It provides a customizable test report.
* It is run tests in parallel or distributed on a Selenium Grid with built-in Selenium Webdriver.
* It allows you to test the functionality from a user’s perspective.
* Sometimes the internal functions of the system work correctly but the user interface doesn’t then GUI testing is good to have in addition to the other types.
* It provides reliable object identification, even for web elements with dynamic IDs.

• Write a scenario of Microwave Owen

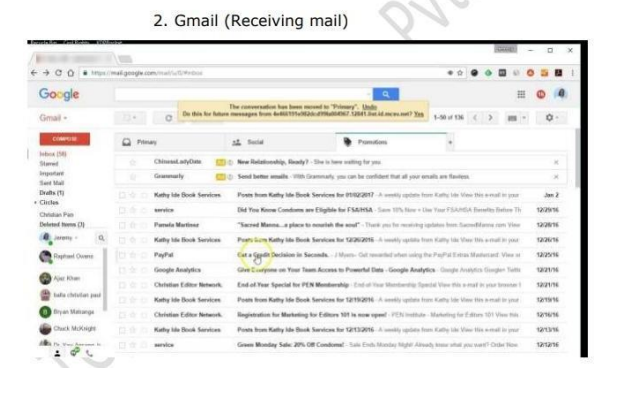
• Write a scenario of Coffee vending Machine

• Write a scenario of chair

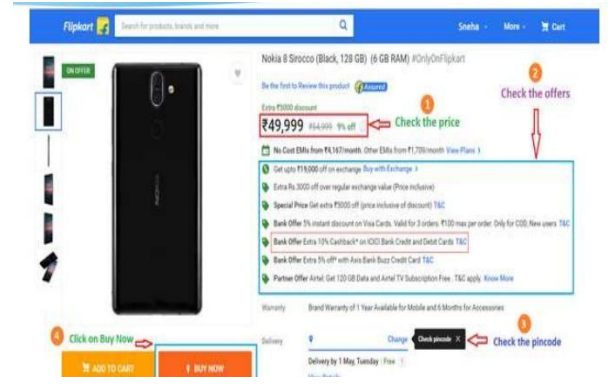
• To Create Scenario (Positive & Negative)

1. facebook Chat on Mobile





3. Online shopping to buy product (flipkart)



Write a Scenario of Wrist Watch

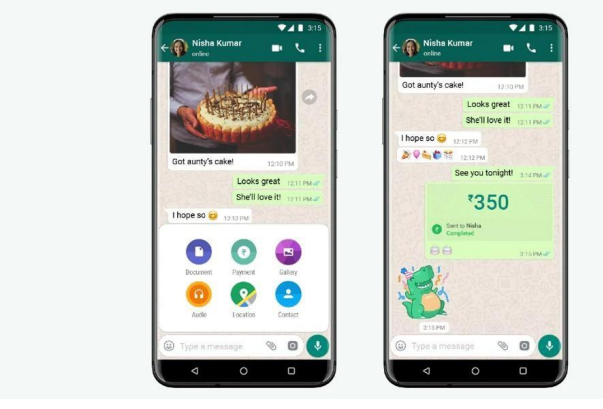
• Write a Scenario of Lift(Elevator)

• Write a Scenario of whatsapp Group (generate group) 

Write a Scenario of instagram ( video call with chat )



Write a Scenario of Whatsapp payment



**Note: All the HLR, Test Cases and Scenario are attached in Excel Sheet.**