1. What is Exploratory Testing?

Experience based techniques are non structured and do not rely on specification documents. This makes them unable to be measured in terms of coverage In experience-based techniques, people’s knowledge, skills and background are of prime importance to the test conditions and test cases.

In exploratory testing tester focuses more on how the software actually works, testers do minimum planning and maximum execution of the software by which they get in depth idea about the software functionality, once the tester starts getting insight into the software he can make decisions to what to test next Exploratory testing is mostly used if the requirements are incomplete and time to release the software is less.

2. 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. A software process should help you keeping the virtual table up-to-date. Simple technique may be quite valuable (naming convention)

Types of Traceability Matrix Forward Traceability

Forward Traceability – Mapping of Requirements to Test cases Backward Traceability – Mapping of Test Cases to Requirement

3. What is Boundary value testing?

Boundary value analysis is a methodology for designing test cases that  
concentrates software testing effort on cases near the limits of valid  
ranges

* Boundary value analysis is a method which refines equivalence  
  partitioning.
* Boundary value analysis generates test cases that highlight errors better  
  than equivalence partitioning.
* The trick is to concentrate software testing efforts at the extreme ends  
  of the equivalence classes.
* At those points when input values change from valid to invalid errors  
  are most likely to occur.
* Boundary Value Analysis (BVA) uses the same analysis of partitions as  
  EP and is usually used in conjunction with EP in test case design

1. What is Equivalence partitioning testing?

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

5. What is Integration Testing?

Integration Testing - Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems Integration Testing is a level of the software testing process where individual units are combined and tested as a group.

The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

Integration testing is done by a specific integration tester or test team. Components may be code modules, operating systems, hardware and even complete systems

There are 2 levels of Integration Testing

1. Component Integration Testing

2. System Integration Testing

6. What determines the level of risk?

Risk A properly designed test that passes, reduces the overall level of Risk in a system Risk – ‘A factor that could result in future negative consequences; usually expressed as impact and likelihood’ When testing does find defects, the Quality of the software system increases when those defects are fixed The Quality of systems can be improved through Lessons learned from previous projects Analysis of root causes of defects found in other projects can lead to Process Improvement.

* Risks are of two types with Example
* Project Risk

Example of Project risk is Senior Team Member leaving the project abruptly. Every risk is assigned likelihood i.e. chance of it occurring, typically on a scale of 1 to 10. Also the impact of that risk is identified on a scale of 1- 10.

But just identifying the risk is not enough. You need to identify mitigation. In this case mitigation could be Knowledge Transfer to other team members & having a buffer tester in place

* Product Risk

Example of product risks would be Flight Reservation system not installing in test environment Mitigation in this case would be conducting a smoke or sanity testing. Accordingly you will make changes in your scope items to include sanity testing.

7. What is Alpha testing?

It is always performed by the developers at the software development site. Sometimes it is also performed by Independent Testing Team. Alpha Testing is not open to the market and public it is conducted for the software application and project. It is always performed in Virtual Environment.

It is always performed within the organization. It is the form of Acceptance Testing. Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers. It comes under the category of both White Box Testing and Black Box Testing.

8. What is beta testing?

It is always performed by the customers at their own site. It is not performed by Independent Testing Team. Beta Testing is always open to the market and public. It is usually conducted for software product. It is performed in Real Time Environment.

It is always performed outside the organization. It is also the form of Acceptance Testing. Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data. It is only a kind of Black Box Testing.

9. 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.

Unit testing frameworks, drivers, stubs and mock or fake objects are used to assist in unit testing. Functional and Non-Functional testing

Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended with debugging tool.

10. What is functional system testing?

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

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

11. What is Non-Functional Testing?

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

An excellent example of non-functional test would be to check how many people can simultaneously login into software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

Non-functional testing includes, but is not limited to, performance  
testing, load testing, stress testing, usability testing, maintainability  
testing, reliability testing and portability testing.

To address this issue, performance testing is carried out to check &  
fine tune system response times. The goal of performance testing is  
to reduce response time to an acceptable level

12. What is GUI Testing?

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

13. What is Adhco testing?

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

* Adhoc testing has
* No Documentation.
* No Test cases.
* No Test Design

As it is not based on any test cases or require documentation or test design so resolving issues that are identified at last becomes very difficult for developers. Sometimes very interesting and unexpected errors or uncommon errors are found which would never have been found in written test cases existed. Actually this Adhoc testing is used in [Acceptance testing](https://www.geeksforgeeks.org/acceptance-testing-software-testing/).

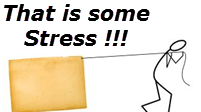
14. What is load testing?

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. 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 falls under the category non-functional testing. Its mainly used for testing the performance of Client/Server and applications which are web based.

15. 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 doesn’t crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.



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

This also called as glass-box testing, clear-box and structural testing. It is based on applications internal code structure. In this, an internal perspective of the system, as well as programming skills are used to design test cases.

In white box testing, the tester analyses the internal architecture of the system as well as the quality of source code on different parameters like code optimization, code coverage, code reusability etc. This testing usually was done at the unit level.

White Box Testing: Testing based on an analysis of the internal structure of the component or system.

1. **Unit Testing**

A unit is the smallest testable part of software.

It usually has one or a few inputs and usually a single output.

• Unit testing conducts on a single program or single module.

• Unit testing is white box testing technique.

• Unit testing is conducted by the developers.

• Unit testing techniques:

• Basis path testing

• Control structure testing

• Conditional coverage

• Loops Coverage

• Mutation Testing

1. Integration Testing

• In Integration Testing, individual software modules are integrated logically and tested as a group.

• Integration testing focuses on checking data communication amongst these modules.

• Integrated Testing is white box testing technique.

• Integrated testing is conducted by the developers.

• Approaches:

• Top Down Approach

• Bottom Up Approach

• Sandwich Approach(Hybrid)

• Stub: Is called by the Module under Test.

• Driver: Calls the Module to be tested.

17. What is black box testing? What are the different black box testing techniques?

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

## Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

* **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
* **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
* **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

## Types of Black Box Testing

* **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
* **Non-functional testing**– This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
* **Regression testing**– Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

18. Mention what are the categories of defects?

Defect from the product or a variance from customer/user expectations. It is a flaw in the software system and has no impact until it affects the user/customer and operational system. Some defects address security or database issues while others may refer to functionality or UI issues.

* 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

* Critical Functionality Defects:

The occurrence of these bugs hampers the crucial functionality of the application. Examples: - Exceptions

* 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

Examples: All JavaScript errors Buttons like Save, Delete, Cancel not performing their intended functions

* 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 for a fix. Examples: Authentication: Accepting an invalid username/password Authorization: Accessibility to pages though permission not given

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

20. Mention what big bang testing is?

**Big Bang Testing** is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

21. 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.

22. When should "Regression Testing" be performed?

Ideally, regression testing should be performed every time changes are made to a system and the code is modified, including adding new features, bug fixes, and performance improvements. It is suitable in scenarios when:

* A new feature or functionality is added to the application
* There is a requirement change
* When defects in the codebase or patches are fixed
* When performance issues are fixed
* When there are environment or configuration changes

23. What are 7 key principles? Explain in detail?

Software Testing is an integral part of SDLC (Software Development Life Cycle). Testing software provides insight into gaps, errors, defects in the product, the quality of the software being developed, and the completeness of specifications in accordance with business, user, and product requirements.

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. All are testing is impossible.

2. 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. That is we must Priorities our testing effort using a Risk Based Approach.

1. Early Testing

Testing activities should start as early as possible in the development life cycle. Remember from our Definition of Testing, that Testing doesn’t start once the code has been written!

4. 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.

Similarly, most operational failures of a system are usually confined to a small number of modules an important consideration in test prioritization!

5. The Pesticide Paradox

If the same tests are repeated over and 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.

1. Testing is Context Dependent

Different kinds of sites are tested differently. Safety – critical software is tested differently from an e-commerce site. Also different industries impose different testing standards

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.

Even after defects have been resolved it may still be unusable and/or does not fulfill the users’ needs and expectations

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

|  |  |  |
| --- | --- | --- |
| Quality Assurance | Quality Control | Testing |
| Sub set of SDLC | Sub set of QA | Sub set of QC |
| Process oriented activities. | Process oriented activities. | Process oriented activities. |
| Preventive activities. | It is a corrective process. | It is a preventive process. |
| Focuses on processes and procedures rather than conducting actual testing on the system | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing. |
| The goal is to keep improving our software development process for the best possible results. | The goal is to find any defects or error in the software and fix them. | It involves running tests and looking at what comes out of them, finding any problems with the software, and ensuring that is does everything it’s supposed to do. |
| Prevent defects | Find and Fix defects | Find and Fix defects |

25. Difference between Smoke and Sanity?

|  |  |
| --- | --- |
| Smoke Testing | Sanity Testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| 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 |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| 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 |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of [Regression Testing](https://www.guru99.com/regression-testing.html) |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

26. 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 compute |

27. Explain types of Performance testing.

* Load

Testing speed of the system while increasing the load gradually till the customer expected number.

* Stress

Testing speed of the system while increasing/reducing the load on the system to check any where its breaking.

* Volume

Check how much volumes of data is able to handle by the system

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

So, we can say that a mistake made by humans during coding is called error, an error found during the testing phase is called a defect, a defect to be resolved by the development team is called a bug and when a build does not meet its specifications then it is termed as failure.

Error

The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The **developers** use the term **error**.

Defect

When the application is not working as per the requirement is knows as **defects**. It is specified as the aberration from the **actual and expected result** of the application or software.

In other words, we can say that the bug announced by the **programmer** and inside the code is called a [**Defect**](https://www.javatpoint.com/defect-or-bug-tracking-tool)**.**

Bug

In [software testing](https://www.javatpoint.com/software-testing-tutorial), a [bug](https://www.javatpoint.com/bug-in-software-testing) is the informal name of defects, which means that software or application is not working as per the requirement. When we have some coding error, it leads a program to its breakdown, which is known as **a bug.** The **test engineers** use the terminology **Bug**.

If a [**QA (Quality Analyst)**](https://www.javatpoint.com/quality-assurance) detect a bug, they can reproduce the bug and record it with the help of the **bug report template**.

Failure

The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

.

|  |
| --- |
|  |
|  |
|  |

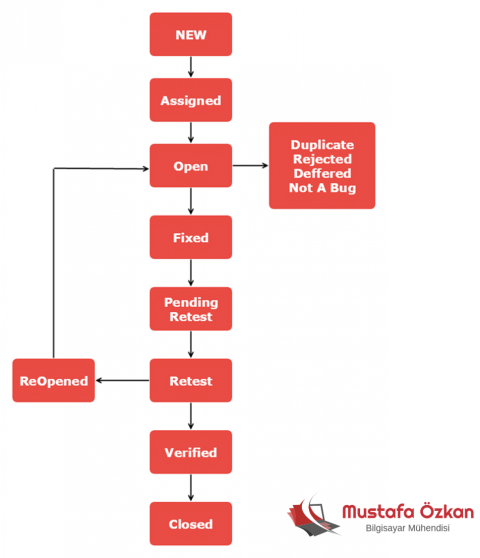
29. Difference between Priority and Severity

|  |  |
| --- | --- |
| Priority | Severity |
| 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 |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| Priority is categorized into three types   * Low * Medium * High | Severity is categorized into five types   * Critical * Major * Moderate * Minor * Cosmetic |

|  |
| --- |
|  |
|  |

30. What is Bug Life Cycle?

As you can see from above diagram, a defect‘s state can be divided into Open or Closed. When a bug reaches one of the Closed or Terminal states, its lifecycle ends. Each state has one or more valid states to move to. This is to ensure that all necessary steps are taken to resolve or investigate that defect. For example, a bug should not move from Submitted state to resolved state without having it open.



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”.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

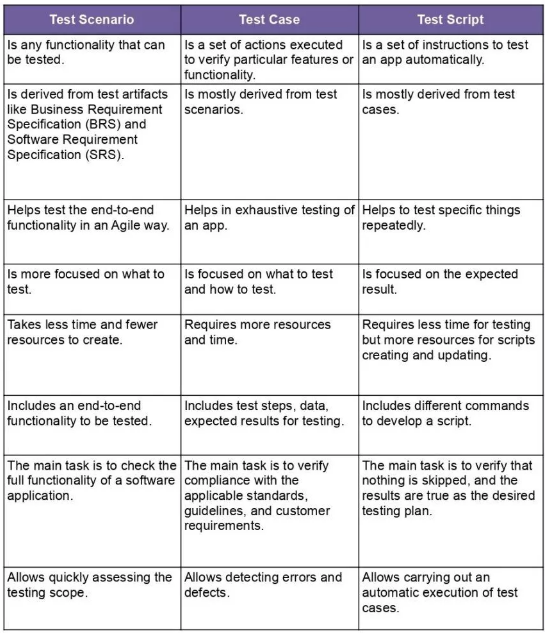
31. Explain the difference between Functional testing and Nonfunctional testing

|  |  |
| --- | --- |
| Functional | Non-Functional |
| Manual testing or automation tools can be used for functional testing | Using tools will be effective for this testing |
| It verifies the operations and actions of an application. | It verifies the behaviour of an application. |
| Functional testing is executed first | Non-functional testing should be performed after functional testing |
| Business requirements are the inputs to functional testing | Performance parameters like speed, scalability are inputs to non-functional testing |
| Functional testing describes what the product does | Non-functional testing describes how good the product works |
| Easy to do manual testing | Tough to do manual testing |
| Types of Functional testing are ∙ Unit Testing ∙ Smoke Testing ∙ Sanity Testing ∙ Integration Testing ∙ White box testing ∙ Black Box testing ∙ User Acceptance testing ∙ Regression Testing | Types of Non-functional testing are ∙ Performance Testing ∙ Load Testing ∙ Volume Testing ∙ Stress Testing ∙ Security Testing ∙ Installation Testing ∙ Penetration Testing ∙ Compatibility Testing ∙ Migration Testing |

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

|  |  |  |
| --- | --- | --- |
| Parameter | SDLC | STLC |
| Origin | Development Life Cycle | Testing Life Cycle |
| Objective | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. | The only objective of the STLC phase is testing. |
| Requirement Gathering | In SDLC the business analyst gathers the requirements and create Development Plan | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan |
| High & Low-Level Design | In SDLC, the development team creates the high and low-level design plans | In STLC, the test analyst creates the Integration Test Plan |
| Coding | The real code is developed, and actual work takes place as per the design documents. | The testing team prepares the test environment and executes them |
| Maintenance | SDLC phase also includes post-deployment supports and updates. | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. |

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



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

A detail of how the test will proceed, who will do the testing, what will be tested, in how much time the test will take place, and to what quality level the test will be performed.

**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.

**What** to test

**Who** do testing? 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.

**Test ware**: Defining the amount, level of detail, structure and templates for the test documentation. Selecting metrics for monitoring and controlling test preparation and execution, defect resolution and risk issues.

**Process**: Setting the level of detail for test procedures in order to provide enough information to support reproducible test preparation and execution.

* 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

35. 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 has to 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.

* ***Priority* can be of following types**:

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

36. What is 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 an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

38. Advantage of Bugzila.

Bugzila is an open-source issue/bug tracking system that allows developers to keep track of outstanding problems with their product. It is written in [Perl](https://www.guru99.com/perl-tutorials.html) and uses MYSQL database.

Bugzila is a [Defect](https://www.guru99.com/defect-management-process.html) tracking tool, however, it can be used as a test management tool as such it can be easily linked with other [Test Case](https://www.guru99.com/test-case.html) management tools like Quality Center, Testlink etc.

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

39. 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. |

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

**1. Kanban**

Kanban is a very popular framework for development in the agile software development methodology. It provides a transparent way of visualizing the tasks and work capacity of a team. It mainly uses physical and digital boards to allow the team members to visualize the current state of the project they are working on. Kanban originated in Toyota in the 1940s. Kanban’s meaning in Japanese is “billboards.” The Kanban board has columns and story cards. The columns are nothing, but workflow states and cards are nothing but a demonstration of the actual task a team member is performing.

**2. SCRUM**

Scrum: SCRUM is an agile development method which concentrates particularly on how to manage tasks within a team based development environment. Basically, Scrum is derived from activity that occurs during rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members).

**3. Extreme Programming**

Extreme Programming (XP) focuses on customer satisfaction through constant development. This methodology emphasizes feedback, communication, and teamwork. Similar to SCRUM, short development cycles or sprints are also present in this framework. It creates a highly efficient and productive environment for software development teams.

**4. Crystal**

Also known as ‘lightweight methodology,’ crystal has less documentation and project review overhead. It doesn’t have a set of rules or any approach to follow, unlike all the previous frameworks we’ve mentioned so far. The Crystal methodology depends on many factors, which are categorized by several different colors.

**5. Dynamic Systems Development**

Dynamic Systems Development Method (DSDM) was developed to address swift software delivery’s need for a standardized industry charter. This methodology believes that project modifications are always expected. It also emphasizes that quality with timely delivery is a must. Those are all based on a business-driven approach.

**6. Feature-driven Development**

This Agile methodology is in contrast to other frameworks like SCRUM and XP. That’s because it centers on strict operations involving domain walkthroughs. It also focuses on design, code, and inspection. Feature-Driven Development (FDD) is centered on the developer and involves turning models into builds at iterations performed every two weeks.

**7. Lean Software Development Methodology**

This Agile methodology type is all about using a holistic approach to give valuable services to customers. Waste reduction is the Lean software development framework’s basic concept.

The principles that guide this methodology include eliminating waste, creating learning, deciding late, and fast delivery. Lean also involves team empowerment, building integrity, and optimizing the whole.

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

|  |  |
| --- | --- |
| Authentication | Authorization |
| Determines whether users are who they claim to be | Determines what users can and cannot access |
| Challenges the user to validate credentials (for example, through passwords, answers to security questions, or facial recognition) | Verifies whether access is allowed through policies and rules |
| Usually done before authorization | Usually done after successful authentication |
| Generally, transmits info through an ID Token | Generally, transmits info through an Access Token |
| Generally governed by the OpenID Connect (OIDC) protocol | Generally governed by the OAuth 2.0 framework |
| 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 |

42. Write a scenario of only Whatsapp chat messages

Positive Test Scenarios

1. Verify that on downloading the Whatsapp application, users can register using a new mobile number.
2. Verify that for a new mobile number user will get a verification code on his mobile and filling in the same verifies the new user account.
3. Check the maximum number of incorrect attempts allowed while filling out the verification code.
4. Verify that registering an existing mobile number for new user account registration is not allowed.
5. Verify that on successful registration all the contacts in the user’s contact directory get imported to the Whatsapp contact list.
6. Verify that the user can set DP and status on Whatsapp.
7. Verify that the user can update the existing DP and Whatsapp status.
8. Verify that the user can send messages to any individual selected from his contact list.
9. Verify that ‘Chats’ window contains all the chat list with DP and name and last message preview of the other person with whom chat was initiated.
10. Verify that clicking a chat in the chat list opens a new window containing all the chats received and sent with the other person.
11. Verify that the user can check the message delivered and read the time for a message in the ‘Message Info’ section.
12. Verify that the user can share or receive contact with the other person.
13. Verify that the user can create a group by adding multiple people from his contact list.
14. Verify that the user can send and receive the message in group chats.
15. Verify that users can send and receive images, audio, video, and emoticons in the chat with individuals.
16. Verify that users can send and receive images, audio, video, and emoticons in group chats.
17. Verify that the user can send and receive chats in the secondary languages available.
18. Verify that users can delete text, images, audio, and video messages within a chat.
19. Verify that users can clear their complete chat history in an individual or group chat.
20. Verify that users can archive chats in an individual or group chat.

Negative Test Scenarios

1. Verify that the user can set a chat wallpaper.
2. Verify that the user sets privacy settings like turning on/off last seen, online status, read receipts, etc.
3. Verify that the user can update notification settings like – notification sound, on/off, and show preview for both group and individual chats.
4. Verify that the user can take the complete chat backup of his chats.
5. Verify that the user can update the phone number that is used by the WhatsApp application.
6. Verify that the user can disable/delete his Whatsapp account.
7. Verify that the user can check data usage by images, audio, video, and documents in WhatsApp chats.

43. Write a Scenario of Pen

## Positive Negative Test Scenarios for Pen

1. Verify the type of pen, whether it is a ballpoint pen, ink pen, or gel pen.
2. Verify that the user is able to 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.

## Negative Test Scenarios for Pen

1. Verify the functioning of a pen at extreme temperatures – much higher and lower than room temperature.
2. Verify the functioning of a pen at extreme altitude.
3. Check the functioning of a pen at zero gravity.
4. Verify the functioning of the pen by applying extreme pressure.
5. Verify the effect of oil and other liquids on the text written with a pen.
6. Check if the user is able to write with a pen when used against gravity i.e. upside down.
7. Verify the functioning of a pen when a user tries to write on unsupported surfaces like glass, plastic, wood, etc.
8. Verify if the pen works normally or not when used after immersing in water or any other liquid for some period of time.

44. Write a Scenario of Pen Stand

1. Verify the type of pen, whether it is a pen stand.
2. Verify that the user is able to write clearly over different types of materials.
3. Check the weight of the pen stand. 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 stand is with a cap or without a cap.
5. Verify the color of the on the pen stand.
6. Check the odor of the pen stands on over a surface.
7. Verify the surfaces over which the pen stand is able to stay smoothly apart from paper e.g. cardboard, rubber surface, etc.
8. Verify that the text written by the pen stand stands 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.

45. Write a Scenario of Door

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

46. Write a Scenario of ATM

* ***Positive Scenario***

**1)** Verify if the card reader is working correctly. A screen should ask you to insert the pin after inserting the valid card.

**2)** Verify if the cash dispenser is working as expected.

**3)** Verify if the receipt printer is working correctly. Which means it can print the data on the paper and the paper comes out properly.

**4)** Verify if the Screen buttons are working correctly. **For touch screen:** Verify if it is operational and working as per the expectations.

**5)** Verify if the text on the screen button is visible clearly.

**6)** Verify the font of the text on the screen buttons.

**7) Verify each number button on the Keypad.**

**8)** Verify the functionality of the Cancel button on the Keypad.

#### Negative Scenario

**1.** Verify the functionality by entering a wrong pin number for 3 or more times.

**2.** Verify the card reader functionality by inserting an expired card.

**3.** Verify the deposit slot functionality by inserting an invalid cheque.

**4.** Verify the cash withdrawal functionality by inserting invalid numbers like 10, 20, 50 etc.

**5.** Verify the cash withdrawal functionality by entering an amount greater than the per day limit,

**6.** Verify the cash withdrawal functionality by entering an amount greater than per transaction limit.

**7.** Verify the cash withdrawal functionality by entering an amount greater than the available balance in the account.

47. When to used Usablity Testing?

**Usability testing** also known as User Experience (UX) testing is a testing method for measuring how easy and user-friendly a software application is.

A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user’s ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.

This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

48. 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. Check all the GUI elements for size, position, width, length and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.

Check you can execute the intended functionality of the application using the GUI Check Error Messages are displayed correctly Check for Clear demarcation of different sections on screen Check Font used in application is readable Check the alignment of the text is proper Check the Color of the font and warning messages is aesthetically pleasing Check that the images have good clarity Check that the images are properly aligned Check the positioning of GUI elements for different screen resolution.

49. Write a scenario of Microwave Owen

1. Verify that the dimensions of the oven are as per the specification provided.
2. Verify that the oven’s material is optimal for its use as an oven and as per the specification.
3. Verify that the oven heats the food at the desired temperature properly.
4. Verify that 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.

50. Write a scenario of Coffee vending Machine

* ***Positive Scenario***

1. Check the indicator lights are displaying correctly when the coffee vending machine is going to switch off or on.
2. Check all the buttons of the coffee vending machine have an image text on them, which indicates what task will be performed if you press the button.
3. Check the complete quantity of copy poured in a single operation, and no coffees are stored in the nozzle area.
4. Check the former up; the coffee vending machine is working as expected.
5. Check the temperature of is served coffee should be the same temperature
6. Check if some ingredients if finished. Then it should display an error message on the LED screen.
7. Check if a button is pressed multiple times; then, the operation should be performed multiple times. If you have pushed the button five times the coffee button, five should be served.
8. Check the time they take to serve coffee should be the same as mentioned in the specification document.

* Negative *Scenario*

1. Check when multiple buttons are pressed simultaneously.
2. Check the performance of the coffee vending machine when the ingredients container’s capacity is exceeded.
3. Check whether the machine gives an error message or works when the boxes are empty.
4. Check whether the machine works if the user enters a higher-value coin.
5. Check whether the machine works if the user enters a lower-value coin.
6. Check whether the machine works if the user enters a foreign currency coin.
7. Check whether the machine works if the user enters a fake coin.
8. Check whether the machine works if the user simultaneously presses the half-cup and full-cup buttons.
9. Check whether the machine works if the user presses all buttons at once.
10. Check whether the machine works if the user places the machine upside down.
11. Check whether the machine works if the user places it horizontally.
12. Check whether the machine stops working if the user tries to open the cover while pouring coffee.

51. Write a scenario of chair

### Test Scenarios For Chair

* Check the material used for making the chair is as per the requirement document.
* Check if the dimension of the chair is as per the specification document.
* Check if the dimension of the weight is as per the specification document.
* Check if the dimension of the height is as per the specification document.
* Check the number of legs of a chair.
* Check the chair backrest option.
* Check whether all legs of the chair on a plane surface are equal.
* Check if the chair is compatible for taking a rest.
* Check whether a human can sit comfortably or not on a chair.
* Check if the chair has an adjustment functionality or not.
* Check the sitting space as per mentioned in the requirement document.
* Check whether the legs of the chair have any wheels or not.
* Check if the chair is good enough to handle a specified load.
* Check what the maximum amount of load the chair is handling is.
* Check the date is stable enough to take any human load.
* Check the color of the table is as per the SRS documents.

### Negative Scenarios for Chair

* Check the balance of the chair with one arm.
* Check the balance of the chair with three legs.
* Check the stress testing of the chair by dropping the Chair down from the practical height.
* Check there nothing is breaking, no damage to the Chair, and the Chair is performed without any issues.
* Check how the Chair is working under different climate environmental conditions.

52. To Create Scenario (Positive & Negative)

1. face book Chat on Mobile

1. Check received messages counts should be displayed on ‘Facebook Message’ icon

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

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

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

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

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

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

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

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

11. Verify that user can search contacts in message box

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

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

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

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

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

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

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

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

20. Verify that spell functionality works fine in chat box

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

22. Verify that user is able to send smiley

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

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

25. Verify that the User is able to share images

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

27. Verify that the User is able to share videos

28. Verify that the User is

2. Gmail Receiving

1. Verify that all the read and unread emails are displayed in the inbox
2. Verify that the recently received email or unread emails are highlighted in bold in the Inbox section.
3. Verify that the recently received email has correct sender‘s name or email id, subject of the email, its preview and date or time.
4. Verifty that the recently received email’s sender’s name or email id, subject of the email, and date or time should be in bold and preivew text shouldn’t be in bold.
5. Verfiy that the attachment icon is displayed next to the preview text of the email, if the email has any attachment.
6. Verify that the Archive, Delete, Mark as read, Snooze options are displaying on hovering the unread email.
7. Verify that the Archive, Delete, Mark as unread, Snooze options are displaying on hovering the read email.
8. Verify that the Email id, Add to contacts, Open detailed view, Send email, Send message, Start video call, Schedule event options are displaying when we hover on the name/email of the read/unread email.
9. Verify that the user is navigated to the email content when clicking on the email in the inbox.
10. Verify that the content of the email is displayed correctly without any formatting issues.
11. Verify that the attachment in the email is downloadable or not.
12. Verify that the attachments can be downloaded as a single zip file.
13. Verify that the attachments can be downloaded individually.
14. Verify that the attachments can be viewable in the browser itself without downlading.
15. Verify that the attachment is downloading in zip format, if the attachment size is more than 1 MB.
16. Verify that the attachments are scanned for viruses once we try to download the file.
17. Verify that the Reply and Forward buttons are displaying in the bottom of the email content.
18. Verify that all the read emails are not highlighted.
19. Verify that unread emails count is displayed beside ‘Inbox’ text in the left sidebar of Gmail.
20. Verify that unread emails count is increased as per the number of new emails we received.
21. Verify that the unread emails count is increased when we mark an email as unread.
22. Verify that the unread emails count is decreased when we mark an email as read or opened.
23. Verify that email recipients in CC are visible to all the users whose emails are present.
24. Verify that email recipients in BCC are not visible to other users in the TO, CC or BCC section.
25. Verify that email can be received from other domains like Hotmail, Outlook, Yahoomail or any other company domains.
26. Online shopping to buy product (flip kart)

53. Write a Scenario of Wrist Watch

Positive Scenario

* Check if the watch looks like a watch.
* Check the watch’s body must not be too thin or too thick.
* Check the body of the watch.
* Check the look and feel must be good.
* Check if the numbers 1 to 12 should present if that’s an analog watch.
* Check if a user increases and decreases the watch strap or chain size.
* Check how long a power cell works in a watch.
* Check how much time it takes to charge the full battery of a smart watch.
* Check how much time it takes to discharge the full battery of a smart watch.

Negative Scenario

* Check if the Watch is working without power or a power cell.
* Check the watch by dropping it from a specific height. After that, check if anything is damaged and if the watch is working as usual without any issues.
* Check if the watch is working or not when the second needle is not working.
* Check if the GPS of the watch is working without a sim and internet.

54. Write a Scenario of Lift(Elevator)

Positive Scenario

* Verify the dimensions
* Verify the type of door is as per the specification
* Verify the type of metal used in the lift interior and exterior
* Verify the capacity  in terms of the total weight
* Verify the buttons in the lift to close and open the door and numbers as per the number of floors
* Verify that lifts move to the particular floor as the button of the floor is clicked.
* Verify that lifts stop when up/down buttons on a particular floor are pressed
* Verify if there is an emergency button to contact officials in case of any mishap
* Verify the performance of the floor – the time taken to go to a floor
* Verify that it doesn’t free-fall in case of a power failure and gets halted on the particular floor.
* Verify lifts are working if the door button is pressed before reaching the destination floor.
* Verify that in case the door is about to close and an object is placed between the doors if the doors since the object again opens or not
* Verify the time duration for which the door remains open by default.
* Verify if the lift interior has proper air ventilation
* Verify lighting in the lift
* Negative Scenario
* Press enter button when we are outside.
* Observe door-opening operations.
* Wait outside for some time when the door is opened.
* When inside, observe door closing operations without pressing the closing button.
* Observe door operations when we middle in the door.
* Check lift is working when the power is off.
* Observe the lift working when the power is low voltage.
* Observe the lift working operations when a heavy load is in the lift.
* Verify door-closing operations when the heavy load is in the lift.
* Check if the power is suddenly off when the lift is in working.
* Add one more person than the specified weight and check the response.
* Add many people regardless of weight and check the volume limitations.
* Create smoke or fire inside the lift and check the response.
* Try pressing the open button while the lift is moving.
* Press the stop button before reaching the specific floor.
* Put small obstacles near the sensors that detect auto-close functionality.

55. Write a Scenario of whatsapp Group (generate group)

* Positive Scenario
* Check that the user can create a Whatsapp group.
* The user can set a name for the created group.
* Check that the user can add and save the group description.
* Verify that the user can make multiple people as group Admin.
* Check that only group admins can add people to the group.
* Is there any option to mute group notifications for some time?
* Is there any option to add people to the group by sharing a link?
* The admin can delete users from this group.
* Check that admin can change settings like only admin can share information in this group, or everyone can share information in this group.
* If a person is removed from the group, he/she will not be able to see any updates.
* Check that user can exit the group by clicking on the exit button or not.
* Verify that the admin can delete the group.
* Check that are you able to create a group without adding any member.
* Negative Scenario
* You can try to send more than 30 images at a time.
* Verify that are you able to send large videos(exceeds limited size) or not.
* Try to send hundreds of messages at the same time to a person and then check the behaviour of this application.
* Try to register with the same number on two different devices at the same time.
* Check that are you able to forward a message to more than five people.
* Try to add more than 256 members in a group.

56 Write a Scenario of instagram (video call with chat)

* Verify the Comparison between Two Captured images of the Same Object in the Same Day Light conditions with the Same Megapixel Camera without any Effect, Zoom, or Flash and with the same Technical features such as PPI, DPI, HDR, Sensor, Resolution, FPS, etc.
* Verify the Comparison between Two same Captured images of the Same Object in the Same Day Light conditions with the Same Megapixel Camera without any Effect, Zoom, or Flash and with different Technical features such as PPI, DPI, HDR, Sensor, Resolution, FPS, etc.
* Verify the Image Detail Comparison between Two same Captured images of the Same Object in the Same Day Light conditions with the Same Megapixel Camera without any Effect, Zoom, or Flash.
* Verify the availability of the Video Recording feature in the Camera.
* Verify the SD quality of the Video Recording feature in the Camera.
* Verify the HD quality of the Video Recording feature in the Camera.
* Verify the Full HD quality of the Video Recording feature in the Camera.
* Verify the Sound Capturing ability during the Video Recording feature in the Camera.
* Verify the Play and Display video with Audio on the screen of SD quality, which is captured by the Video Recording feature of the Camera.
* Verify the Play and Display video with Audio on the HD quality screen, which is captured by the Video Recording feature of the Camera.
* Verify the Play and Display video with Audio on the screen of Full HD quality, captured by the

57 Write a Scenario of Whatsapp payment

1. Open a chat with the person or business you wish to send money to.
2. Tap the **Attach** (paperclip) icon, and select **Payment**.
3. Key in the amount you wish to send.
4. Authenticate the payment using the 6-digit Facebook Pay PIN or your fingerprint.
5. The money will be transferred to the recipient, and you will both receive notifications in the chat window.