*Software Testing Assignment*

Module – 2 (Manual Testing)

1. What is Exploratory Testing?

Exploratory Testing is a type of software testing in which the tester is free to select any possible methodology to test the software. It is an unscripted approach to software testing. In exploratory testing, software developers use their learning, knowledge, skills, and abilities to test the software developed by themselves. Exploratory testing checks the functionality and operations of the software as well as identify the functional and technical faults in it. Exploratory testing aims to optimize and improve the software in every possible way. The exploratory testing technique combines the experience of testers with a structured approach to testing. It is often performed as a black box testing technique.

1. What is traceability matrix?

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship. It is used to track the requirements and to check the current project requirements are met.

1. What is Boundary value testing?

Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions. The behaviour at the edge of the equivalence partition is more likely to be incorrect than the behaviour within the partition, so boundaries are an area where testing is likely to yield defects. It checks for the input values near the boundary that have a higher chance of error. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

* A boundary value for a valid partition is a valid boundary value.
* A boundary value for an invalid partition is an invalid boundary value.
* For each variable we check-
* Minimum value.
* Just above the minimum.
* Nominal Value.
* Just below Max value.
* Max value.

1. What is Equivalence partitioning testing?

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

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

1. What is Integration testing?

Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.

1. What determines the level of risk?

Risk management is the process of identifying, analyzing, evaluating, and addressing potential risks in an organization.

Risk management involves the following four stages:

* Identification: Identify potential risks that could negatively impact the organization’s objectives and goals.
* Analysis: Analyze the identified risks to determine their likelihood and potential impact.
* Evaluation: Evaluate the risks based on their likelihood and potential impact to determine which one’s merit attention.
* Treatment: Address the identified risks using appropriate strategies and techniques to mitigate or eliminate them.

1. What is Alpha testing?

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

Phases of Alpha Testing

* Planning
* Preparation
* Execution
* Evaluation
* Reporting
* Closure

1. What is beta testing?

Beta testing is the process of testing a software product or service in a real-world environment before its official release. It is an essential step in the software development lifecycle as it helps identify bugs and errors that may have been missed during the development process.

Types of Beta Testing

* Traditional Beta testing
* Public Beta Testing
* Technical Beta Testing
* Focused Beta Testing
* Post-release Beta Testing

1. 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.

1. 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.

Functional Testing Types

* Unit testing
* Smoke testing
* User Acceptance
* Integration Testing
* Regression testing
* Localization
* Globalization
* Interoperability

1. What is Non-Functional Testing?

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

Types of Non-Functional Testing

* Performance Testing
* Load Testing
* Compatibility Testing
* Usability Testing
* Stress Testing
* Maintainability Testing
* Scalability Testing
* Security Testing

1. What is 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.

1. What is 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 approach. That’s why Adhoc testing is a type of Unstructured Software Testing.

Adhoc testing has –

* No Documentation
* No Test cases
* No Test Design

Types of Adhoc Testing:

* Buddy Testing
* Pair Testing
* Monkey Testing

1. What is load testing?

Load testing is a type of Performance Testing that determines the performance of a system, software product, or software application under real-life-based load conditions. Load testing determines the behaviour of the application when multiple users use it at the same time.

During load testing, various scenarios are simulated to test the system’s behavior under different load conditions. This can include simulating a high number of concurrent users, simulating numerous requests, and simulating heavy network traffic. The system’s performance is then measured and analyzed to identify any bottlenecks or issues that may occur.

1. What is stress Testing?

Stress Testing is a software testing technique that determines the robustness of software by testing beyond the limits of normal operation. Stress testing is particularly important for critical software but is used for all types of software. Stress testing emphasizes robustness, availability, and error handling under a heavy load rather than what is correct behavior under normal situations.

Stress testing is defined as a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions. It even tests beyond the normal operating point and analyses how the system works under extreme conditions. Stress testing is performed to ensure that the system would not crash under crunch situations. Stress testing is also known as Endurance Testing.

Types of Stress Testing:

* Transactional Stress Testing
* Systematic Stress Testing
* Server-client Stress Testing
* Product Stress Testing
* Application Stress Testing

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

White box testing techniques analyse the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

Types of White Box Testing

* Unit Testing
* Static Analysis
* Dynamic Analysis
* Statement Coverage
* Branch Testing
* Path Testing
* Loop Testing

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

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

The testing techniques are:

* Equivalent Partitioning
* Boundary Value Analysis
* Decision Tables
* State Transition Testing
* Use Case Testing
* Syntax or Pattern Testing

1. Mention what are the categories of defects?

Software Defect is some kind of error, flaw or some kind of mistake from the development team which prevent the software from the smooth working. It directly affects software quality; software quality is something how smooth and reliable your software is. Smoothness and reliability are how less defects your software have.

Categories of defects:

* Errors of commissions
* Errors of omissions
* Errors of clarity
* Error of speed and capacity

1. Mention what big bang testing is?

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

1. What is the purpose of exit criteria?

Exit criteria are the defined requirements within software testing that must be met in order to determine that testing has been completed. These conditions are typically defined by engineering or test leadership to ensure quality standards are met.

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

1. When should "Regression Testing" be performed?

Regression testing is a type of software testing that ensures that existing functionality works as expected after making changes to the code, such as adding new features, fixing bugs, or improving performance. Regression testing helps to avoid introducing new defects or breaking existing features while modifying the software. In this article, you will learn when you should perform regression testing and what factors to consider when planning and executing it.

It depends on the scope and complexity of the changes, the type and level of testing, the available resources and time, and the testing strategy and methodology.

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

* Testing shows presence of defects
* Exhaustive testing is not possible
* Early testing
* Defect clustering
* Pesticide paradox
* Testing is context dependent
* Absence of errors fallacy

1. Testing shows presence of defects:

The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn’t talk about the absence of defects. Software testing can ensure that defects are present but it cannot prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.

1. Exhaustive testing is not possible:

It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case, then it will take more cost, effort, etc., which is impractical.

1. Early testing:

To find the defect in the software, early test activity shall be started. The defect detected in the early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

1. Defect clustering:

In a project, a small number of modules can contain most of the defects. The Pareto Principle for software testing states that 80% of software defects come from 20% of modules.

1. Pesticide paradox:

Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

1. Testing is context dependent:

The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing. For example, the testing of the e-commerce site is different from the testing of the Android application.

1. Absence of errors fallacy:

If a built software is 99% bug-free but does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfil all the customer requirements.

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

The quality of a product, application, or website plays a vital role in its success. While developers build the functionalities, testers’ job is to determine the software’s quality and effectiveness. However, the definition of quality can be subjective, making it crucial to establish methods and processes to verify the product’s quality for both those involved in creating it and its users.

In software development, the terms ‘Testing,’ ‘Quality Assurance,’ and ‘Quality Control’ are often thrown around interchangeably. But it’s important to understand that Quality Control vs Quality Assurance vs Testing are distinct concepts that form different parts of the quality management cycle. In this blog, we delve into the significance of these three terms and explore how they can be utilized to enhance the quality of a product.

This blog discusses the differences between Quality Control, Quality Assurance, and Testing. These are all essential parts of ensuring software works well, so understanding how they work together is vital if you want a great product. We’ll look at these processes and explain what makes them unique. If you’re into software testing or engineering or are just curious about software development, this blog will be super helpful! Let’s start exploring the world of Quality Control vs Quality Assurance vs Testing!

1. 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.

1. Difference between verification and Validation

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfils the requirements that we have. Verification is static testing.

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.

1. Explain types of Performance testing

* Load testing:

Load testing is a type of testing which involves evaluating the performance of the system under the expected workload. A typical load test includes determining the response time, throughput, error rate, etc during the course of the load test.

* Stress testing:

Stress testing is a type of performance testing where we evaluate the application’s performance at a load much higher than the expected load. Another aspect of the stress testing is to determine the break-point of the application, the point at which the application fails to respond in the correct manner.

* Endurance testing

Endurance testing is also known as ‘Soak Testing’. It is done to determine if the system can sustain the continuous expected load for a long duration. Issues like memory leakage are found with endurance testing.

* Spike testing

In spike testing, we analyse the behavior of the system on suddenly increasing the number of users. It also involves checking if the application is able to recover after the sudden burst of users.

* Volume testing

The volume testing is performed by feeding the application with a high volume of data. The application can be tested with a large amount of data inserted in the database or by providing a large file to the application for processing. Using volume testing, we can identify the bottleneck in the application with a high volume of data.

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

A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which results in a bug.

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

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

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

1. Difference between Priority and Severity

Bug Severity or Defect Severity in testing is a degree of impact a bug or a Defect has on the software application under test. A higher effect of bug/defect on system functionality will lead to a higher severity level. A Quality Assurance engineer usually determines the severity level of a bug/defect.

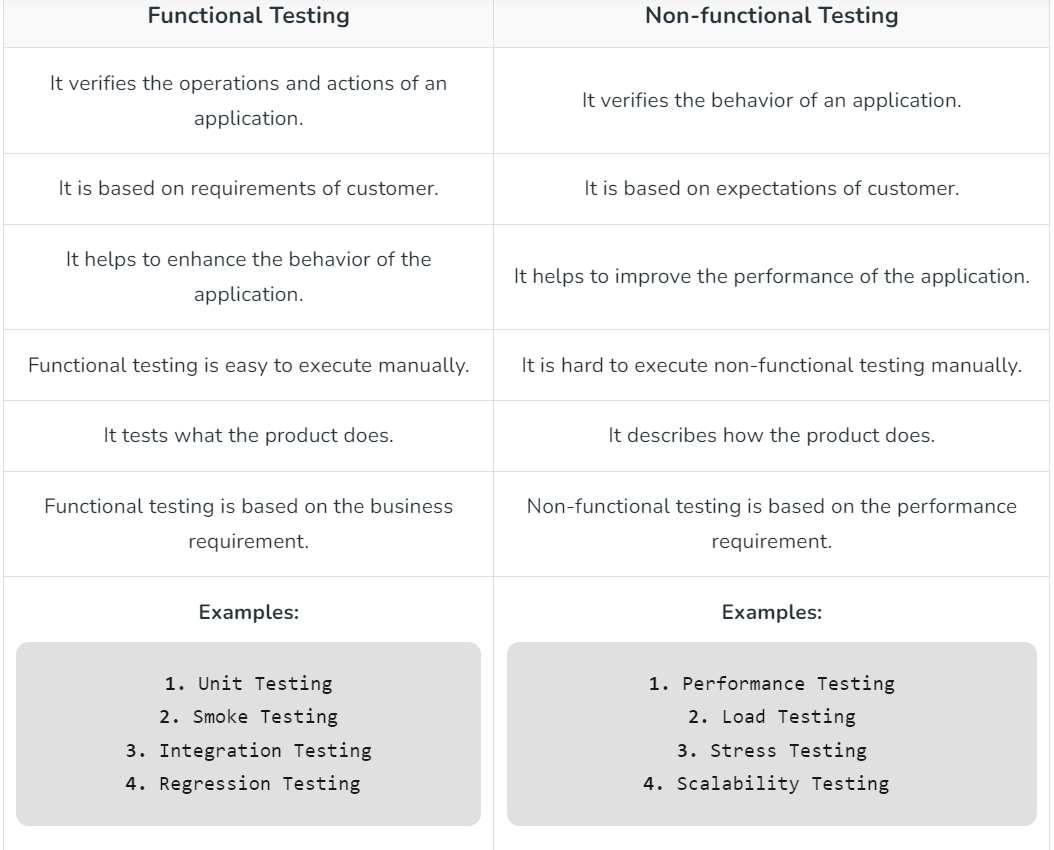
Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved. Defects that leave the software system unusable are given higher priority over defects that cause a small functionality of the software to fail.

1. What is Bug Life Cycle?

Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

Bug Status in defect life cycle is the present state from which the defect or a bug is currently undergoing. The goal of defect status is to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.

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



1. To create HLR & Test Case of
2. (Instagram, Facebook) only first page

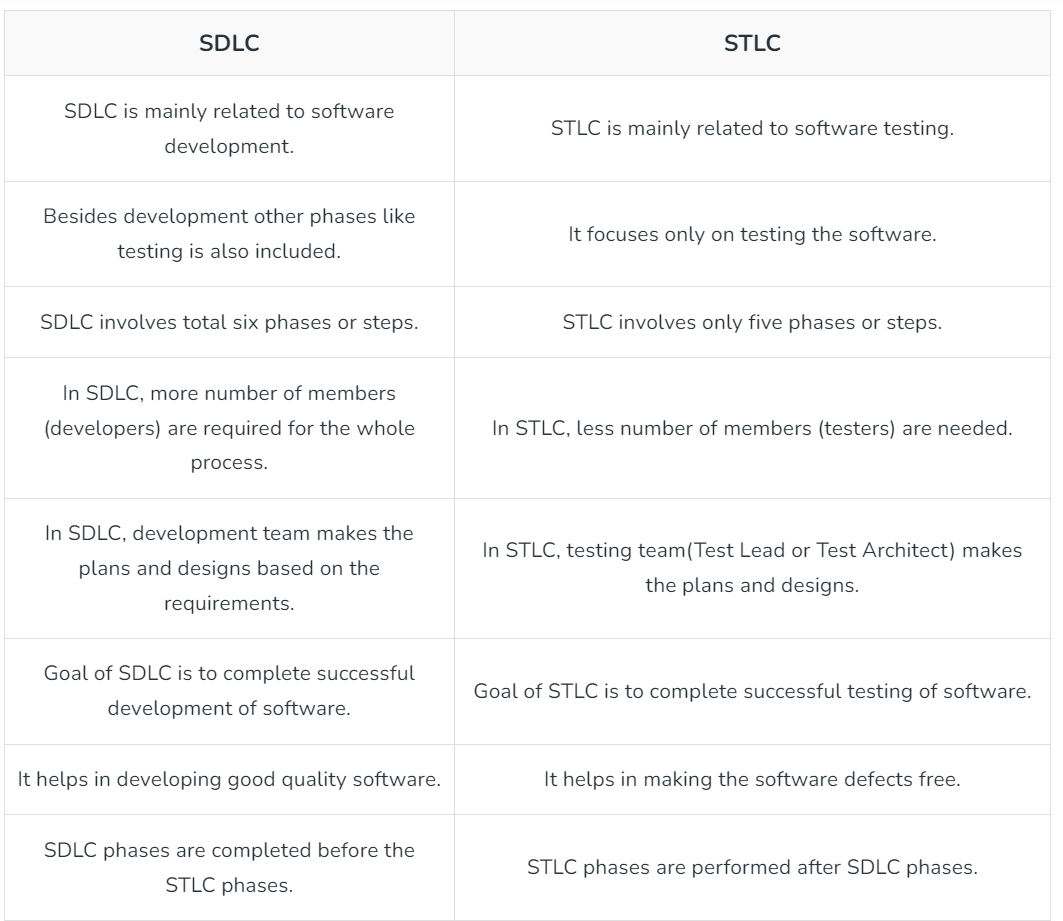
* **HLR of Instagram Login Page -** <https://github.com/niralilad2/ST_work/blob/main/HLR%20of%20Instagram%20Login%20Page.xlsx>
* **Test Case of Instagram Login Page -** <https://github.com/niralilad2/ST_work/blob/main/TEST%20CASE%20-%20INSTAGRAM%20LoginPage.xlsx>

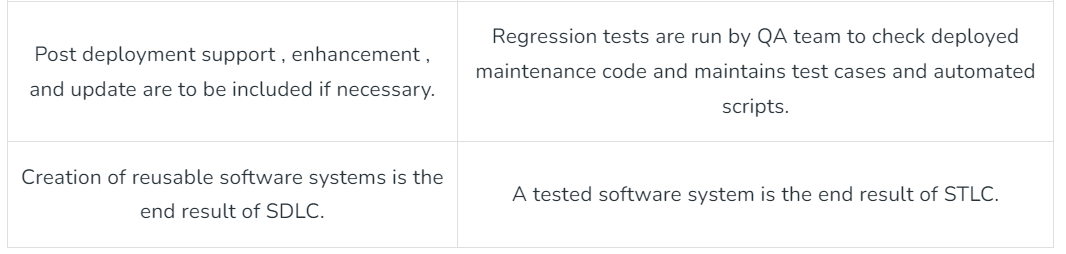
1. Facebook Login Page: <https://www.facebook.com/>

* **HLR of Facebook Login Page –** <https://github.com/niralilad2/ST_work/blob/main/HLR%20of%20Facebook%20Login%20Page.xlsx>
* **Test Case of Facebook Login Page -** <https://github.com/niralilad2/ST_work/blob/main/TEST%20CASE%20-%20FACEBOOK%20Login%20Page.xlsx>

1. What is the difference between the STLC (Software Testing Life Cycle) and SDLC

(Software Development Life Cycle)?





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

**Test Scripts**

A test script is the most detailed way to document software testing. It typically has 'steps' in the form of code that should be performed manually. These scripts also include expected results for each step, for example, 'Click the Apply button', with an example result of 'A form opens'. Test scripts are written in programming languages, such as Java, Python, Ruby, and are short programs used to test discrete parts of the software system. In other words, test scripts are automated sets of steps that have to be executed by the tester.

A few things should be considered before going all-in with test scripts. Software projects change often; pages get redesigned and user navigation also changes. This is the reason test management software should be used to update the scripts to match the new versions. Using testing software saves time and is also cost effective in the longer run.

Secondly, by using testing software, scripts can be made more versatile by testing the same condition using different steps and different data. To achieve this, however, testers need to be more intuitive and creative when writing scripts.

**Test Cases**

A test case is a documented set of preconditions (prerequisites), procedures (inputs/actions) and postconditions (expected results) which a tester uses to determine whether a system under test satisfies requirements or works correctly. Test cases have a great impact on the testing phase. Writing test cases is almost as important as the testing process itself.

The activity of writing test cases helps you think through the details and ensures you're approaching the tests from as many angles as possible. The long-term value of having test cases is that anyone can go in and retest using the test case. Test cases are powerful artefacts that are also beneficial for future teammates, as well as a good source of knowing how a system and particular feature works.

**Test Scenarios**

Test scenario is the least detailed and high-level type of documentation. A test scenario is a description of an objective a user might face when using the program. They cover an end-to-end functionality which is to be tested. An example might be 'Test that the user can successfully log out by closing the program.' So, this scenario will require testing in a few different ways to ensure the scenario has been entirely covered.

Test cases are derived from test scenarios, whereas test scenarios are derived from test artefacts such as Software Requirement Specification (SRS) and Business Requirement Specification (BRS) documents. Test scenarios ensure that the business processes and flows are as per the functional requirements. They also serve as a quick tool to determine the most critical end-to-end transactions or the real use of the application.

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

A test plan is a document that outlines the objectives, scope, approach, resources, and schedule for a testing project. It defines the strategy for how testing will be conducted, including what will be tested, who will perform the tests, the test environment, and the criteria for success. The purpose of a test plan is to ensure that testing is systematic, thorough, and aligned with the project's goals and requirements.

A comprehensive test plan typically includes the following information:

* Introduction: Provides an overview of the document and its purpose.
* Objectives: Clearly states the goals and objectives of the testing effort.
* Scope: Defines the boundaries of the testing, including what will and will not be tested.
* Approach: Describes the overall strategy for testing, including methodologies, techniques, and tools to be used.
* Test Items: Lists the specific components or features to be tested.
* Features to be tested: Describes the features or functionalities to be tested, often with reference to requirements or specifications.
* Features not to be tested: Identifies any features or functionalities that will not be tested and the reasons for exclusion.
* Test Environment: Specifies the hardware, software, and other resources needed for testing.
* Test Deliverables: Lists the documents, reports, and other deliverables expected from the testing process.
* Test Schedule: Outlines the timeline for testing activities, including milestones and deadlines.
* Test Team: Identifies the roles and responsibilities of individuals involved in testing.
* Risks and Assumptions: Identifies potential risks to the testing process and any assumptions made during planning.
* Dependencies: Lists any dependencies or constraints that may impact testing activities.
* Approvals: Specifies the individuals or stakeholders who must review and approve the test plan.
* Appendices: Includes any additional information or reference materials relevant to the testing effort.

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

* Conducting meetings: The team conducts frequent meetings for 10-15 minutes daily, and they think that conducting frequent meetings will be Agile. However, only the following meetings will not be Agile.
* Requirements changing anytime: Requirements can be changed at any time, then it is not Agile. For example, a client wants to add some new features and want the changes to be updated at the same time, then this will not be Agile.
* Unstructured development: Suppose you are not following any plan and you are working on Ad-hoc basis then it is not Agile wherein Ad-hoc testing, testers randomly test the application without following any documents and test design techniques.
* No documentation: If the company does not make the documentation, then it is not Agile.
* What is agile? The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams’ cycle through a process of planning, executing, and evaluating. Agile is a philosophy, i.e., a set of values and principles to make a decision for developing software.
* Agile is based on the iterative-incremental model. In an incremental model, we create the system in increments, where each increment is developed and tested individually.
* What are values?

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

* Working software, over comprehensive documentation: Documentation is needed, but working software is much needed. Agile is not saying that documentation is not needed, but working software is much needed. For example, you have 20-page documents, but you do not have a single prototype of the software. In such a case, the client will not be happy because, in the end, the client needs a document.
* Customer collaboration, over contract negotiation: Contract negotiation is important as they make the budget of software, but for example, if you stuck with the requirements or process, then do not go for a contract which we have negotiated. You need to interact with the customer, gather their requirements.
* Responding to change, over following a plan
* In the waterfall model, everything is planned, i.e., at what time, each phase will be completed. Sometimes you need to implement the new requirements in the middle of the software, so you need to be versatile to make changes in the software.
* Agile Principles: Customer satisfaction through early and continuous software delivery: Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
* Accommodate changing requirements throughout the development process – The ability to avoid delays when a requirement or feature request changes.
* Frequent delivery of working software – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
* Collaboration between the business stakeholders and developers throughout the project – Better decisions are made when the business and technical team are aligned.
* Support, trust, and motivate the people involved – Motivated teams are more likely to deliver their best work than unhappy teams.
* Enable face-to-face interactions – Communication is more successful When development teams are co-located.
* Working software is the primary measure of progress – Delivering functional software to the customer is the ultimate factor that measures progress.
* Agile processes to support a consistent development pace – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
* Attention to technical detail and design enhances agility – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
* Simplicity – Develop just enough to get the job done for right now.
* Self-organizing teams encourage great architectures, requirements, and designs – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
* Regular reflections on how to become more effective – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.
* Methodologies:
* Scrum: A framework for managing and controlling iterative and incremental projects, emphasizing communication, collaboration, and delivering working software in short iterations called sprints.
* Kanban: A visual management method that aims to optimize the flow of work by visualizing the workflow, limiting work in progress, and continuously improving the process.
* Extreme Programming (XP): Emphasizes principles such as rapid feedback, continuous testing, continuous integration, and simple design to deliver high-quality software incrementally.
* Lean Software Development: Adapted from Lean manufacturing principles, it focuses on eliminating waste, amplifying learning, empowering teams, and delivering as fast as possible.
* Crystal: A family of methodologies that prioritize communication and teamwork, with different variants suited for various project sizes and contexts.
* Dynamic Systems Development Method (DSDM): Emphasizes active user involvement, frequent delivery of products, and adapting to changing requirement.

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

* Authorization and authentication are both essential aspects of web testing, but they serve different purposes:
* Authentication: Authentication is the process of verifying the identity of a user or entity attempting to access a system or resource. In web testing, this typically involves validating that a user is who they claim to be by verifying their credentials, such as username and password. Common authentication methods include username/password, biometric authentication, single sign-on (SSO), and multi-factor authentication (MFA).
* Authorization: Authorization, on the other hand, determines what actions or resources a user is allowed to access after they have been
* authenticated. It is the process of granting or denying access rights and privileges to authenticated users based on their identity and the permissions associated with their account or role. In web testing, authorization testing involves verifying that users can only access the resources and perform the actions they are authorized to, based on their role or privileges.
* Common problems faced in web testing include:
* Security vulnerabilities: Web applications are often targets for malicious attacks, so security testing is crucial to identify and address vulnerabilities such as injection flaws, cross-site scripting (XSS), cross-site request forgery (CSRF), and insecure direct object references.
* Compatibility issues: Web applications need to work across various browsers, devices, and operating systems. Compatibility testing ensures that the application functions correctly and displays properly across different environments.
* Performance issues: Poor performance, slow loading times, and high server response times can negatively impact user experience and lead to decreased engagement and customer dissatisfaction. Performance testing helps identify and address bottlenecks, scalability issues, and other performance-related problems.
* Usability concerns: Usability testing evaluates the user interface, navigation, and overall user experience of the web application to ensure it is intuitive, user-friendly, and meets the needs of its target audience.
* Functionality bugs: Functional testing verifies that the web application functions correctly according to its specifications and requirements.
* Testers need to identify and report any bugs, glitches, or inconsistencies in the application's functionality.

1. To create HLR & Test Case of Web Based (WhatsApp web, Instagram) 1. WhatsApp Web: https://web.whatsapp.com/

**HLR of Web Based of WhatsApp Web –** <https://github.com/niralilad2/ST_work/blob/main/HLR%20of%20WHATSAPP%20WEB.xlsx>

**Test Case of Web Based of WhatsApp Web -** <https://github.com/niralilad2/ST_work/blob/main/TEST%20CASE%20-%20WHATSAPP%20WEB.xlsx>

**HLR of Web Based of Instagram –** <https://github.com/niralilad2/ST_work/blob/main/HLR%20of%20Instagram%20Web%20Login%20Page.xlsx>

**Test Case of Web Based of Instagram -** <https://github.com/niralilad2/ST_work/blob/main/TEST%20CASE%20-%20INSTAGRAM%20WEB%20LoginPage.xlsx>

1. To create HLR and Test Case on this Link. <https://artoftesting.com/>

**HLR of Art of Testing -** <https://github.com/niralilad2/ST_work/blob/main/HLR%20of%20ART%20OF%20TESTING.xlsx>

**Test Case of Art of Testing -** <https://github.com/niralilad2/ST_work/blob/main/TEST%20CASE%20of%20ART%20OF%20TESTING.xlsx>

1. Write a scenario of only WhatsApp chat messages

• Verify that the Chat window contains the entire chat list.

• Verify that the Chat window displays the contact numbers whose numbers not saved in mobile.

• Verify that Chat window displayed with all contacts with DP or without DP.

• Verify that the Chat window is displayed on the group chat list.

• Verify that the Chat window displays the last updated chatting time.

• Verify that the Chat window displays the name of all contacts on the chat window.

• Verify that clicking on one Chat contact then a new window should open with history.

• Verify that the user can see all delivered and received messages.

• Verify that the user can see the read or send time of messages.

• Verify that the user can send and receive text messages in the individual chat box.

• Verify that user can send and receive documents in the individual chat box.

• Verify that user can send and receive photos in individual chat box.

• Verify that user can send and receive videos in individual chat box.

• Verify that user can send and receive audio in individual chat box.

• Verify that user can send and receive emotions icons in individual chat box.

• Verify that user can send and receive Contacts in individual chat box.

• Verify that user can send and receive Location in individual chat box.

• Verify that user can send and receive GIF in individual chat box.

• Verify that user can send and receive Stickers in individual chat box.

• Verify that user can delete text, video, audio, locations, documents in individual chat box.

• Verify that user can send recorded voice mail in individual chat box.

• Verify that user can delete the entire chat history in individual chat box.

• Verify that user is able to see contact details in individual chat box.

• Verify that user is able to share images, links, and documents from media in individual chat box.

• Verify that user is able to search specific chat history using search option in individual chat box.

• Verify that user is able to video call in individual chat box.

• Verify that user is able to voice call in individual chat box.

• Verify that user is able to mute the individuals in individual chat box.

• Verify that user is able to share images, links, and documents from media in individual chat box.

• Verify that user is able to search specific chat history using search option in individual chat box.

• Verify that user is able to change the wallpaper.

• Verify that users have options like Report, Block, Clear Chat, Export Chat and Add Shortcut.

1. Write a Scenario of Pen

• Verify that user is able to hold pen comfortably.

• Verify that the pen is writing smoothly or not.

• Verify that pen is not making any type of sound while writing.

• Verify that the ink flow of the pen is smooth or not. There should not be any break in ink flow.

•Verify that the quality of the material of pen is good or bad.

• Verify that the company name of pen is visible clearly or not.

• Verify that the design on pen or the branding of the pen is not removed easily.

• Verify that the line drawn by the pen is according to the mentioned specifications.

• Verify that the ink colour of the pen should be consistent from start to end.

• Verify that the pen is able to write on different types of paper. (Smooth, Glossy, rough, thick, thin.)

• Verify that the ink of pen is waterproof or not.

• Verify that the ink of pen is dried quickly on paper or not.

• Verify that other refills fit the pen or not.

• Verify that the pen shouldn't have any sharp edges and corners.

• Verify that the body or pen and the ink is made up of nontoxic material or not.

• Verify that pen is working normally or not after putting the pen in water for some time.

• Verify that pen is working normally or not after dropping the pen from some height.

• Verify that ink spelled or not after dropping the pen from some height.

•Verify that pen is able to write when left open for some time without pen cap.

•Verify that nib of pen is able to withstand the pressure while writing.

• Verify that the pen is working under high pressure or not.

• Verify that pen is working under high temperature or not.

• Verify that pen is working under low pressure or not.

• Verify that p t pen is working where gravity is zero.

1. Write a Scenario of Pen Stand

• Verify that the pen stand is sturdy and stable.

• Verify any visible defects or damages in the material.

• Verify that maximum number of pens or other stationery items that can be accommodated in the pen stand.

• Verify if the material is resistant to scratches, stains, or other forms of damage that may occur during regular use

• Verify Evaluate the weight distribution of the pen stand by placing pens of varying weights and sizes in different compartments.

• Verify Ensure that the pen stand remains stable and balanced, even when loaded with items unevenly.

• Verify the compatibility of the pen stand with different desk surfaces, including wood, glass, or plastic.

• Verify Ensure that the pen stand does not scratch or damage the desk surface during normal use.

• Verify Assess the ease of access to pens or other items stored in the pen stand.

Determine how easy it is to clean the pen stand, considering factors such as material type and design intricacies.

• Verify the pen stand is resistant to water or if it requires special cleaning instructions to maintain its appearance.

• Verify if the base of the pen stand provides adequate support to prevent tipping or wobbing.

• Verify the pen stand arrives without damages or defects that may have occurred during transit.

• Consider user reviews and ratings to understand overall satisfaction with the pen stand.

1. Write a Scenario of Door

• Verify the door is single door or bi-folded door.

• Verify the door opens inwards or outwards.

• Verify that the dimension of the doors is as per the specifications.

• Verify that the material used in the door body and its parts is as per the specifications.

• Verify that color of the door is as specified.

• Verify if the door is sliding door or rotating door.

• Verify the position, quality and strength of hinges.

• Verify the type of locks in the door.

• Verify the number of locks in the door interior side or exterior side.

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

• Verify if the door is having stopper or not.

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

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

• Verify the door condition when used extensively with water.

• Verify the door condition in different climatic conditions- temperature, humidity etc.

• Verify the amount of force- pull or push required to open or close the door.

• Verify the handle to open/close the door, can be used by various size of human hands

• Verify that we are able to lock and unlock the door

• Verify if someone knocking the door we are able to listen or not

• Verify that someone people knocking the door we are able to see

• Verify the door is water proof or not.

1. Write a Scenario of ATM

• Verify that power backup should be present at ATM.

• Verify that card reader should be present.

• Verify that receipt printer should be present and working.

• Verify that cash dispenser is working as expected.

• Verify that the key pad should be working and covered.

• Verify that buttons are displayed on screen of ATM machine.

• Verify the font of text on the screen, it should be clearly visible.

• Verify that when card in inserted in ATM, pin should be asked from user.

• Verify that when user enters incorrect pin for a particular number of times, the card is blocked.

•Verify that when user enters correct pin, the user details should be displayed on ATM screen.

• Verify that ATM machine asks to user for the amount to be withdrawn.

• Verify that if use enters amount greater than daily withdraw limit, error message is displayed.

• Verify that if doesn't enter amount in round off digits, error message is displayed.

• Verify that if user enters valid amount, the exact amount of cash should be dispensed from ATM machine.

• Verify that how much time is taken in a transaction.

• Verify how much time is taken by system to logout user.

• Verify that user is able to use card of another bank on the ATM.

• Verify that message is displayed when the cash in ATM is finished.

• Verify that correct message is displayed after the transaction.

• Verify that user is presented with an option to select language of operation.

• Verify that pin is displayed in masked format.

• Verify that error message is displayed when entered amount is greater than account balance.

• Verify that session is timeout is no activity is performed for a particular time.

• Verify that the user is not allowed to exceed one transaction limit amount.

• Verify that the user is not allowed to exceed the one-day transaction limit amount.

• Verify the functionality by entering a wrong pin number for a particular number of times.

• Verify the card ATM machine functionality by inserting an expired card.

•Verify the cash withdrawal functionality by entering invalid amount such as 10, 20, 50.

•Verify the ATM machine functionality by entering amount greater than available balance. y entering amour

• Verify the ATM machine functionality by entering amount greater than per day and per transaction limit.

1. When to used Usability Testing?

Web Based Testing, Desktop Based. Mobile based & Game based Testing:

All fields on a page (For Example, text box, radio options, drop-down lists) should be aligned properly

The user should not be able to type in drop-down select lists.

Tab and Shift +Tab order should work properly.

All buttons on a page should be accessible by keyboard shortcuts and the user should be able to perform all operations using a keyboard

All buttons on a page should be accessible by keyboard shortcuts and the user should be able to perform all operations using a keyboard,

All pages should have a title

Confirmation messages should be displayed before performing any update or delete operation

Hourglass should be displayed when the application is busy

Page text should be left-justified.

The user should be able to select only one radio option and any combination for checkboxes

- Goal of Usability Testing

Effectiveness of the system

Efficiency

Accuracy

User Friendliness

HOW MANY USERS DO YOU NEED?

5(five) users are enough to uncover 80% of usability problems. Some researchers suggest other numbers. The truth is, the actual number of users required depends on the complexity of the given application and your usability goals. Increase in usability participant's results into increased cost, planning, participant management and data analysis.

But as a general guideline, if you on a small budget and interested in DIY usability testing 5 is a good number to start with. If budget is not a constraint its best consult experienced professionals to determine number of users.

1. 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 barstool bar, menu bar, dialog boxes and windows etc.

WHAT DO YOU CHECK IN GUI TESTING?

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.

- Approach of GUI Testing

MANUAL BASED TESTING

Under this approach, graphical screens are checked manually by testers in conformance with the requirements stated in business requirements document.

RECORD AND REPLAY

GUI testing can be done using automation tools. This is done in 2 parts. During Record, test steps are captured into the automation tool. During playback, the recorded test steps are executed on the Application under Test. Example of such tools - QTP.

MODEL BASED TESTING

A model is a graphical description of system's behaviour. It helps us to understand and predict the system behaviour. Models help in a generation of efficient test cases using the system requirements.

- GUI Testing Examples:

Web Based Testing & Desktop Based Testing:

The scrollbar should be enabled only when necessary.

Font size, style, and color for headline, description text, labels, infield data, and grid info should be standard as specified in SRS.

The description text box should be multi-lined.

Enough space should be provided between field labels, columns, rows, error messages, etc.

- Mobile Based Testing:

If mobile is in every orientation mode so display image, video properly.

Every app will display in responsive type.

Alignment should be applied properly of every field.

- Game Based Testing:

Game infra design will showing properly

Game points or score will display proper with its background color.

Game sound manage with its background effect

Can be also conducted in advance of designing page layouts or navigation menus.

1. Write a scenario of Microwave Owen

• Verify that type of microwave oven like solo, grill or convection

• Verify that the company name is properly displayed or not

• Verify that the Brand logo is properly displayed on the microwave oven or not

• Verify that size of the microwave oven

• Verify that colour of the microwave oven

• Verify that material of the microwave oven

• Verify that capacity of the microwave oven

• Verify that the compact design of the microwave oven

• Verify that glass is turn able or not

• Verify that weight of the microwave oven

• Verify that dimensions of the microwave oven

• Verify that voltage of the microwave oven

• Verify that batteries are required or not

• Verify that all buttons are properly worked or not

• Verify that food is properly reheating or not

• Verify that food is grilled properly or not

• Verify that the digital displayed screen should be properly visible to users

• Verify that oven's door is properly opened and get closed

• Verify that different kind of food at different temperature

• Verify that different kind of containers

• Verify that temperature functionality is properly working or not

• Verify that the alarm sound system is properly working or not

• Verify that glass rotation speed is as expected

• Verify oven's condition when it runs for specific hours

• Verify that disconnecting power while cooking is in progress

1. Write a scenario of Coffee vending Machine

• Verify that the dimension of the coffee machine is as per the specification.

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

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

• Verify the input mechanism for coffee ingredients-milk, water, coffee beans/powder.

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

• Verify the power/voltage requirements of the machine.

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

• Verify that coffee should not leak when not in operation.

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

• Verify that the digital display displays correct information.

• Verify the machine can be switched on and off using the power buttons.

• Verify for the indicator lights when the machine is switched on-off.

• Verify that the functioning of all the buttons work properly when pressed.

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

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

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

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

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

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

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

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

• Verify the amount of time the machine takes to serve a single serving of coffee.

• Verify the performance of the machine when used continuously until the ingredients run out of the requirements.

• Verify the functioning of the coffee machine when two/multiple buttons are pressed simultaneously.

• Verify the functioning of coffee machine with a lesser or higher voltage than required.

• Verify the functioning of the coffee machine if the ingredient container’s capacity is exceeded.

• Verify for the cup holder dimension as per specification/ or market standard

• Verify for guarantee & warranty of the machine, in case provided.

1. Write a scenario of chair

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

•Verify that material used in making the chair-wood, plastic etc.

•Verify that chair’s leg is level to the floor.

•Verify that usability of the chair as an office chair, normal household chair.

•Verify there is back support in the chair.

•Verify there is support for hands in the chair.

•Verify the paint’s type and color.

•Verify if the chair’s material is brittle or not.

•Verify cushion is provided with chair or not.

•Verify condition when washed with water or effect of water on chair.

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

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

•Verify the height of the chair’s seat from floor.

•Verify whether distance between all legs it should be same.

•Verify by design because there is a chair with 3 legs or modern design when two and two are different distance

•Verify that wheels are needed specified in chair.

•Verify the wheels are turning 360 degrees or not.

•Verify the wheels are oiled well or not.

•Verify that chair can be upgraded (different back) or add the wheels

•Verify chair can be stored with other chairs on top/bottom

•Verify that strength and material of the chair

•Verify that the Main Function check stability

•Verify edged of chair is needed to be in round shape or sharp

1. To Create Scenario (Positive & Negative) Gmail (Receiving mail)

• Verify that a newly received email is displayed as highlighted in the Inbox section.

• Verify that a newly received email has correctly displayed sender email Id or name, mail subject and mail body (trimmed to a single line).

• Verify that the user receives the email in their inbox.

• Verify that on clicking the newly received email, the user is navigated to email content.

• Verify that the email contents are correctly displayed with the desired source formatting.

• Verify that any attachments are attached to the email and are downloadable.

• Verify that the attachments are scanned for viruses before download.

• Verify that all the emails marked as read are not highlighted.

• Verify that all the emails read as well as unread have a mail read time appended at the end on the email list displayed in the inbox section.

• Verify that count of unread emails is displayed alongside ‘Inbox’ text in the left sidebar of Gmail.

• Verify that unread email count increases by one on receiving a new email.

• Verify that unread email count decreases by one on reading an email (marking an email as read).

• Verify that email recipients in cc are visible to all users.

• Verify that email recipients in bcc are not visible to the user.

• Verify that all received emails get piled up in the ‘Inbox’ section and get deleted in cyclic fashion based on the size availability.

• Verify that email can be received from non-Gmail email Ids like – yahoo, Hotmail etc.

1. Online shopping to buy product (Flipkart)

• Verify the initiation of the buy flow.

• Verify the accuracy of product details.

• Verify responsiveness to changes in quantity.

• Verify the behaviour with out-of-stock items.

• Verify the adding products from different categories.

• Verify the accuracy and function of applied discounts.

• Verify that on the product page, the user can select the desired attribute of the product e.g. size, color, etc.

• Verify that users can add products to the Wishlist.

• Verify that the user can see the previously added products on the cart page, after signing in to the application.

• Verify that the user can successfully buy more than one product that were added to his/her cart.

• Verify that the user cannot add more than the available inventory of the product.

• Verify that the limit to the number of products a user can buy is working correctly. Also, an error message gets displayed, preventing the user from buying more than the limit.

• Verify that the delivery can be declined during checkout for the places where shipping is not available.

• Verify that the Cash on Delivery option of payment is working fine.

• Verify that the different prepaid methods of payments are working fine.

• Verify that product return functionality works correctly

• Verify the visibility and accuracy of delivery options.

• Verify responsiveness of the payment selection step.

• Verify the behaviour with saved addresses.

• Verify the accuracy of order summary details.

• Verify responsiveness to changes in user location.

• Verify the redirection to the Order Confirmation page.

• Verify the cancelling the payment process midway.

1. Write a Scenario of Wrist Watch

•Verify the type of watch – analog or digital.

•Verify the correctness time displayed by the second, minute, and hour hand of the watch.

•Verify the digital display for hours, minutes, and seconds is correctly displayed.

•Verify the material of the watch and its strap.

•Verify the shape of the dial is as per specification.

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

•Verify the weight of the watch.

•Verify the watch is waterproof or not.

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

•Verify the watch is having a date and day display or not.

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

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

•Verify the second hand of the watch makes ticking sound or not.

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

•Verify the clock is having stopwatch, timers, and alarm functionality or not.

•Verify the format of the watch 12 hours or 24 hours.

•Verify the watch comes with any guarantee or warranty.

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

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

•Verify the battery requirement of the watch.

•Verify belt or chain used is comfortable or not and its length

•Verify chain material and belt for damage.

1. Write a Scenario of Lift (Elevator)

• Verify the maximum passenger capacity of lift.

• Verify the maximum weight capacity of lift.

• Verify the Height, Width and length of lift.

• Verify the type of material used in lift interior and exterior.

• Verify the type of door of lift.

• Verify that buttons are present according to the number of floors in building.

• Verify that buttons to open and close lift door should be present.

• Verify that the buttons should be self-explanatory.

• Verify that blind person should be able to use the buttons.

• Verify that the controls are guided by a voice message.

• Verify that emergency button and contact details should be present in lift.

• Verify the performance of lift, the time taken to go a particular floor.

• Verify that fan is present in lift or not.

• Verify that light is present in lift or not.

• Verify that lift moves to particular floor when the floor button is pressed.

• Verify that lift comes to floor is up/down button are pressed at a particular floor.

• Verify that the door should not be opened if open is door button is pressed in between two floors.

• Verify the behaviour of lift in case of power failure, lift should not free fall or should not stop in between two floors.

• Verify that lift door closes or not if an object is placed in between the door, it should not close.

• Verify the time for which the door remains open at floor.

• Verify that door should not open while the lift is in motion.

• Verify that backup mechanism should be present in case of power loss.

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

• Verify that in case of capacity limit is reached users are prompted with warning alert.

• Verify that inside lift user is prompted with current floor and direction information the lift is moving towards.

• Verify the behaviour of lift when the capacity of lift exceeds.

Verify the behaviour of lift when there is smoke or fire inside the lift.

• Verify the behaviour by pressing open button while lift is moving.

• Verify the behaviour by pressing stop button before reaching the specific floor.

1. Write a Scenario of WhatsApp Group (generate group)

• Verify that user is able to create a new or not.

• Verify that user is able to add multiple contacts from contact list.

• Verify that user is able to insert group name and select image for DP.

• Verify that user is able to add and remove contacts from group.

• Verify that user is able to delete a group.

• Verify that user can send and receive text messages in group.

• Verify that user can send and receive documents in group chat box.

• Verify that user can send and receive photos in group chat box.

• Verify that user can send and receive videos in in group chat box.

• Verify that user can send and receive audio in in group chat box.

• Verify that user can send and receive emotions icons in in group chat box.

• Verify that user can send and receive Contacts in group chat box.

• Verify that user can send and receive Location in group chat box.

• Verify that user can send and receive GIF in in group chat box.

• Verify that user can send and receive Stickers in group chat box.

• Verify that user can delete text, video, audio, locations, documents in group chat box.

• Verify that user can send recorded voice mail in group chat box.

• Verify that user is able make multiple video call in group chat box.

• Verify that user is able to see the group contact information from Group Info in group chat box.

• Verify that user is able to shared images, links, and documents from Group Media in group chat box.

• Verify that user is able to search specific chat history using search option in group chat box.

• Verify that user is able to mute the group in group chat box.

• Verify that users have options like Report, Block, Clear Chat, Export Chat and Add Shortcut.

1. Write a Scenario of WhatsApp payment

• Verify that users can register for WhatsApp Payment.

• Verifying the user's phone number for WhatsApp payment.

• Verify that users can link their bank account to WhatsApp for transactions.

• Verify that sending money from one user to another within the WhatsApp application.

• Verify that users can select a contact from their address book to send money to.

• Verify that users can enter the amount they wish to send and add a note if necessary.

• Verify that users can review the transaction details before confirming.

• Verify that receiving money from another user.

• Verify that users receive a notification when they have received money.

• Verify that the transaction details are visible within the chat interface.

• Verify that users can view their transaction history.

• Verify that transaction history includes details such as date, time, amount, sender/receiver, and transaction status (completed, pending, failed, etc.).

• Verify that security features such as PIN or biometric authentication before making a payment.

• Verify that users receive confirmation messages for each transaction, including details about the recipient and amount sent.

• Verify that adding a bank account for transactions.

• Verify that users can add multiple bank accounts if supported.

• Verify that users can edit or remove bank accounts as needed.

• Verify that there are limits on the amount of money that can be sent or received per transaction and per day

• Verify that performance of the payment feature under various network conditions (3G, 4G, 5G, Wi-Fi).

• Verify that transactions are processed promptly and reliably without delays or errors.