MODULE :- 2

MANUAL TESTING

1) WHAT IS EXPLORATORY TESTING :-

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

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

2) 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.

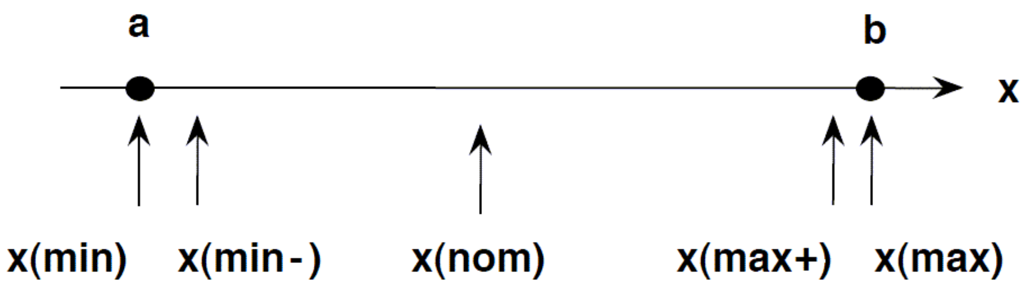
Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

3) WHAT IS BOUNDARY VALUE TESTING :-

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

- So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.
- The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



- In Boundary Testing, Equivalence Class Partitioning plays a good role
- Boundary Testing comes after the Equivalence Class Partitioning.

4) **WHAT IS EQUIVALENCE PARTITIONING TESTING :-**

Equivalence Partitioning or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

- It divides the input data of software into different equivalence data classes.
- You can apply this technique, where there is a range in the input field.

5) **WHAT IS INTEGRATION TESTING :-**

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

6) **WHAT DETERMINES THE LEVEL OF RISK :-**

Several factors determine the risk levels in risk management, including the likelihood and potential impact of risks, the organization's risk appetite, industry regulations and standards, and the organization's risk management experience and maturity.

Another important factor that determines risk levels in risk management is the external environment. This includes factors such as economic conditions, political stability, and natural disasters. These external factors can greatly impact the likelihood and potential impact of risks, and organizations must take them into account when assessing and managing risks.

Additionally, the effectiveness of an organization's risk management strategy and processes can also impact risk levels. A well-designed and implemented risk management framework can help to identify and mitigate risks more effectively, reducing overall risk levels. On the other hand, a poorly designed or implemented framework can lead to increased risk levels and potential negative consequences for the organization.

7) **WHAT IS ALPHA TESTING :-**

Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of acceptance testing, The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.
This testing is referred to as an alpha testing only because it is done early on, near the end of the development of the software, and before Beta Testing. Check Differences between Alpha testing and Beta testing.

Alpha testing is typically performed by in-house software engineers or QA staff. It is the final testing stage before the software is released into the real world.

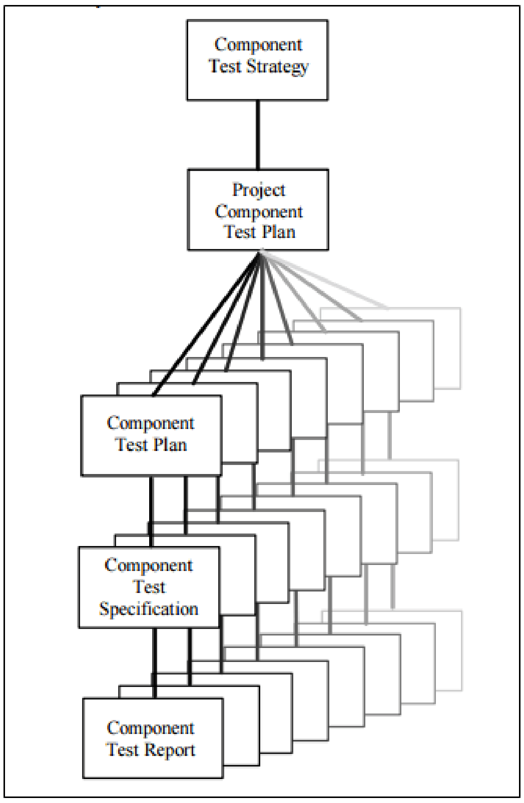
8) **WHAT IS COMPONENT TESTING :-**

Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It's also referred to as Module Testing when it is

viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.

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

As per the below diagram, there will be a test strategy and test plan for component testing. Where each and every part of the software or application is considered individually. For each of this component a Test scenario will be defined, which will be further brought down into a High Level Test Cases -> Low Level detailed Test Cases with Prerequisites.



The usage of the term "Component Testing" varies from domain to domain and organization to organization.

9) **WHAT IS BETA TESTING :-**

Beta Testing is performed by "real users" of the software application in "real environment" and it can be considered as a form of external User Acceptance Testing. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer's environment.

Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

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

12) **WHAT IS GUI TESTING ?**

There are two types of interfaces for a computer application. Command Line Interface is where you type text and computer responds to that command. GUI stands for Graphical User Interface where you interact with the computer using images rather than text.

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 ADHOC TESTING ?**

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called Error Guessing. Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

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

14) **WHAT IS LOAD TESTING ?**

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

Load Testing image

This testing usually identifies –

The maximum operating capacity of an application

Determine whether the current infrastructure is sufficient to run the application

Sustainability of application with respect to peak user load

Number of concurrent users that an application can support, and scalability to allow more users to access it.

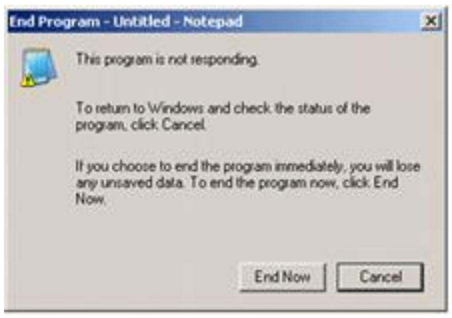
It is a type of non-functional testing. In Software Engineering, Load testing is commonly used for the Client/Server, Web-based applications – both Intranet and Internet.

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.

In Software Engineering, Stress Testing is also known as Endurance Testing. Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity. A most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks. It also checks whether the system demonstrates effective error management under extreme conditions.

The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives 'Not Responded' error message.


End Program - Untitled - Notepad

This program is not responding.

To return to Windows and check the status of the program, click Cancel.

If you choose to end the program immediately, you will lose any unsaved data. To end the program now, click End Now.

16) WHAT IS WHITE BOX TESTING AND LIST TYPES OF WHITE BOX TESTING ?

White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

➢ **TYPES OF WHITE BOX TESTING**

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below —

- **Unit Testing**: It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
- **Testing for Memory Leaks**: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

Apart from the above, a few testing types are part of both black box and white box testing. They are listed below

- **White Box Penetration Testing**: In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats.
- **White Box Mutation Testing**: Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

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.

Input
Black Box
Output

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

➢ **BLACK BOX TESTING TECHNIQUES :-**

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

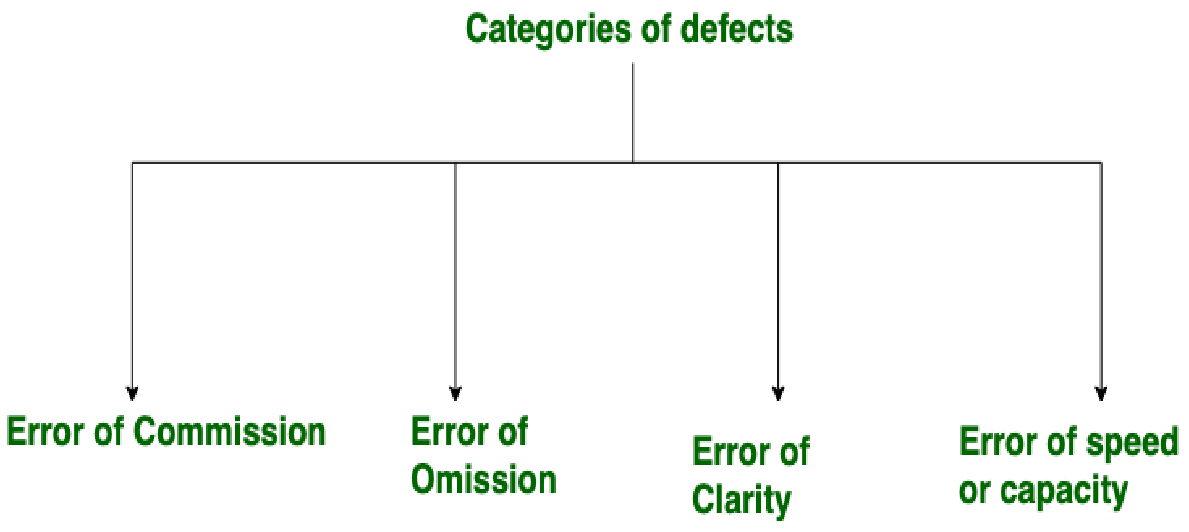
- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.

- **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.

18) MENTION WHAT ARE THE CATEGORIES OF DEFECTS ?

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

Categories of defects: Categories of defects are: Errors of commissions, Errors of omissions, Errors of clarity, and Error of speed and capacity.



These are explained as following below.

1. **Error of Commission:** Commission means instruction or some kind of command given. Now the error in commission means the error in made in command or instruction. For example, suppose I wrote a loop which I was trying to run 10 times but I command it to run more than 10 times by mistake this is the error of commission.
2. **Errors of Omissions:** As name is already describing error of omission is some thing which happens accidentally. Omission word means something left out or executed. Practical most common example of this error is suppose we make a function in programming open its bracket but forget to close at the end.
3. **Error of Clarity:** The most common error in the natural languages. This error happens due to miss understanding between the developer and client. It travels most of the time from the requirements to the software.
4. **Error of Speed or Capacity:** The name of the error is itself enough i think to tell about it this error. Your software is working fine but not working in the required time this is the error of speed. When it comes to capacity it can be relevant to memory. For example, a small integer is declared where the long integer was required.

19) **MENTION WHAT BIGBANG 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.

Advantages:

Convenient for small systems.

1)

Disadvantages:

Fault Localization is difficult.

Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.

Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.

Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

20) **WHAT IS THE PURPOSE OF EXIT CRITERIA ?**

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.

Creating exit criteria helps:

- Align your teams on a common definition of test completion
- Ensure your product meets completion standards before entering the next stage, which avoids costly project delays
- Create clear parameters for test engineers to evaluate software

21) **WHEN SHOULD ' REGRESSION TESTING ' BE PERFORMED ?**

Here are the scenarios when you can apply the regression testing process.

- **New functionality is added to the application:** This happens when new features or modules are created in an app or a website. The regression is performed to see if the existing features are working as usual with the introduction of the new feature.
- **In case of change requirement:** When any significant change occurs in the system, regression testing is used. This test is done to check if these shifts have affected features that were there.
- **After a defect is fixed:** The developers perform regression testing after fixing a bug in any functionality. This is done to determine if the changes made while fixing the bug have affected other related existing features.
- **Once the performance issue is fixed:** After fixing any performance issues, the regression testing process is triggered to see if it has affected other existing functional tests.
- **While integrating with a new external system:** End-to-end regression testing process is required whenever the product integrates with a new external system.

22) **WHAT IS 7 KEY PRINCIPLES ? EXPLAIN IN DETAIL.**

- **Principles of Software Testing**

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy

1) Exhaustive testing is not possible

Yes! Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.

And the million dollar question is, how do you determine this risk?

To answer this let's do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle Defect Clustering

2) Defect Clustering

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

4) Testing shows a presence of defects

Hence, testing principle states that – Testing talks about the presence of defects and don't talk about the absence of defects. i.e. Software 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.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

This leads us to our next principle, which states that- Absence of Error

5) Absence of Error – fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user's needs & requirements.

To solve this problem, the next principle of testing states that Early Testing

6) Early Testing

Early Testing – Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software's are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

23) **DIFFERENCE BETWEEN QA V/S QC V/S TESTER .**

QA **(QUALITY ASSURANCE)**	**QC** **(QUALITY CONTROL)**	**TESTER**
Focus on process and procedure.	Focus on product.	Focus on actual testing.
Process oriented activity.	Product oriented activity.	Product oriented activity.
Preventive activity.	It is a corrective process.	It is a preventive process.
Subset of SDLC.	Subset of QA.	Subset of QC.
Verifies the quality.	Validates the quality.	Validates the quality.
The Whole Project Team involved.	Testing Team involved.	Testing Team involved.
Makes sure the right things are being done.	Makes sure the things are doing right.	Evaluates the results of done things.
When : throughout the process.	Before the release.	At the testing stage or along with the development process.
Proactive process.	Reactive process.	Reactive process.

24) **DIFFERENCE BETWEEN SMOKE AND SANITY .**

SMOKE TESTING	SANITY TESTING
Smoke testing is done to assure that the acute functionalities of program is working fine.	Sanity testing is done to check the bugs have been fixed after the build.
Smoke testing is also called subset of acceptance testing.	Sanity testing is also called subset of regression testing.
Smoke testing is documented.	Sanity testing isn't documented.
Smoke testing is performed by either developers or testers.	Sanity testing is normally performed by testers.
Smoke testing may be stable or unstable.	Sanity testing is stable.
Smoke testing is scripted.	Sanity testing is usually not scripted.
Smoke testing is done to measure the stability of the system/product by performing testing.	Sanity testing is done to measure the rationality of the system/product by performing testing.
Smoke testing is used to test all over function of the system/product.	Sanity testing is used in the case of only modified or defect functions of system/products.
Smoke testing can be performed either manually or by using automation tools.	Sanity testing is commonly executed manually, not by using any automation approach.
Smoke testing is performed when new product is built.	Sanity testing is conducted after the completion of regression testing.
It includes all the system's essential basic functionality.	It includes only those modules where change in code is made.
Smoke Testing firstly performs on the initial build. Smoke testing is done first.	Sanity Testing is done on stable builds or for the introduced new features in the software.
Smoke testing can be carried out either way-manually or automatically.	Without using test cases or scripts sanity testing can be carried out.

There is end-to-end system verification done in smoke testing.	A specific component gets verified in sanity testing.
In the smoke testing process, the software build could be stable or unstable.	During sanity testing, the software build is comparatively stable.
For every new build release smoke testing is carried out.	Sanity testing is carried out when in-depth testing is not possible because of short time.

25) **DIFFERENCE BETWEEN VERIFICATION AND VALIDATION.**

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.
Verification refers to the set of activities that ensure software correctly implements the specific function.	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
After a valid and complete specification the verification starts.	Validation begins as soon as project starts.
Verification is for prevention of errors.	Validation is for detection of errors.
Verification is also termed as white box testing or static testing as work product goes through reviews.	Validation can be termed as black box testing or dynamic testing as work product is executed.
Verification finds about 50 to 60% of the defects.	Validation finds about 20 to 30% of the defects.
Verification is based on the opinion of reviewer and may change from person to person.	Validation is based on the fact and is often stable.

Verification is about process, standard and guideline.	Validation is about the product.

26) **EXPLAIN TYPES OF PERFORMANCE TESTING .**

Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under a particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and is also known as *“Perf Testing”***.**

- **Types of Performance Testing**

Load testing – checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.

- **Stress testing** – involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- **Endurance testing** – is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing** – tests the software’s reaction to sudden large spikes in the load generated by users.
- **Volume testing** – Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
- **Scalability testing** – The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

27) **WHAT IS ERROR, DEFECT, BUG AND FAILURE ?**

- **What is an Error?**

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.

- Errors are generated due to wrong logic, syntax, or loop that can impact the end-user experience.
- It is calculated by differentiating between the expected results and the actual results.
- It raises due to several reasons like design issues, coding issues, or system specification issues and leads to issues in the application.

- **What is a Defect?**
  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.

  - The defect is an issue in application coding that can affect the whole program.
  - It represents the efficiency and inability of the application to meet the criteria and prevent the software from performing the desired work.
  - The defect can arise when a developer makes major or minor mistakes during the development phase.

➢ **What is a Bug?**

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. It is now that the Automation/ Manual Test Engineers describe this situation as a bug.

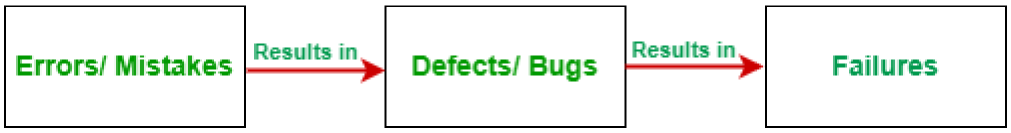
- A bug once detected can be reproduced with the help of standard bug-reporting templates.
- Major bugs are treated as prioritized and urgent especially when there is a risk of user dissatisfaction.
- The most common type of bug is a crash.
- Typos are also bugs that seem tiny but are capable of creating disastrous results.

➢ **What is a Failure?**

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.

- Failure can happen due to human errors or can also be caused intentionally in the system by an individual.
- It is a term that comes after the production stage of the software.
- It can be identified in the application when the defective part is executed.

A simple diagram depicting Bug vs Defect vs Fault vs Failure:



28) DIFFERENCE BETWEEN PRIORITY AND SEVERITY.

Features	Severity	Priority
Definition	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.
Purpose	Severity means how severe the defect is affecting the functionality.	Priority means how fast the defect has to be fixed.
Relation	Severity is related to the quality standard.	Priority is related to scheduling to resolve the problem.

Categories	Severity is divided into 4 categories: Critical Major Medium Low	Priority is divided into 3 categories: Low Medium High
Who decides defects?	The testing engineer decides the severity level of the defect.	The product manager decides the priorities of defects.
Value	Its value is objective.	Its value is subjective.
Value change	Its value doesn't change from time to time.	Its value changes from time to time.
Association	It is associated with functionality or standards.	It is associated with scheduling.
Indication	It indicates the seriousness of the bug in the product functionality.	It indicates how soon the bug should be fixed.
Driving factor	It is driven by functionality	It is driven by business value.
Based On	It is based on the technical aspect of the product.	It is based on the customer's requirements.

29) WHAT IS BUG LIFE CYCLE ?

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

- The journey of the Defect Cycle varies from organization to organization and also from project to project because development procedures and platforms as well as testing methods and testing tools differ depending upon organizations and projects.
- The number of states that a defect goes through also varies depending upon the different tools used and processes followed during the testing of software.
- The objective of the defect lifecycle is to easily coordinate and communicate the current status of the defect and thus help to make the defect-fixing process efficient.

30) EXPLAIN THE DIFFERENCE BETWEEN FUNCTIONAL TESTING AND NON FUNCTIONAL TESTING.

PARAMETERS	FUNCTIONAL TESTING	NON FUNCTIONAL TESTING
EXECUTION	It is performed before non-functional testing.	It is performed after the functional testing.
FOCUS AREA	It is based on customer's requirements.	It focusses on customer's expectation.
REQUIREMENT	It is easy to define functional requirements.	It is difficult to define the requirements for non-functional testing.

USAGE	Helps to validate the behavior of the application.	Helps to validate the performance of the application.
OBJECTIVE	Carried out to validate software actions.	It is done to validate the performance of the software.
REQUIREMENTS	Functional testing is carried out using the functional specification.	This kind of testing is carried out by performance specifications
MANUAL TESTING	Functional testing is easy to execute by manual testing.	It's very hard to perform non-functional testing manually.
FUNCTIONALITY	It describes what the product does.	It describes how the product works.
EXAMPLE TEST CASE	Check login functionality.	The dashboard should load in 2 seconds.
TESTING TYPES	Examples of Functional Testing Types • Unit testing • Smoke testing • User Acceptance • Integration Testing • Regression testing • Localization • Globalization • Interoperability	Examples of Non-functional Testing Types • Performance Testing • Volume Testing • Scalability • Usability Testing • Load Testing • Stress Testing • Compliance Testing • Portability Testing • Disaster Recover Testing

31) **WHAT IS THE DIFFERENCE BETWEEN SDLC AND STLC.**

SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE)	STLC **(SOFTWARE TESTING LIFE CYCLE)**
SDLC is mainly related to software development.	STLC is mainly related to software testing.
Besides development other phases like testing is also included.	It focuses only on testing the software.
SDLC involves total six phases or steps.	STLC involves only five phases or steps.
In SDLC, more number of members (developers) are required for the whole process.	In STLC, less number of members (testers) are needed.

In SDLC, development team makes the plans and designs based on the requirements.	In STLC, testing team(Test Lead or Test Architect) makes the plans and designs.
Goal of SDLC is to complete successful development of software.	Goal of STLC is to complete successful testing of software.
It helps in developing good quality software.	It helps in making the software defects free.
SDLC phases are completed before the STLC phases.	STLC phases are performed after SDLC phases.
Post deployment support , enhancement , and update are to be included if necessary.	Regression tests are run by QA team to check deployed maintenance code and maintains test cases and automated scripts.
Creation of reusable software systems is the end result of SDLC.	A tested software system is the end result of STLC.

32) **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.

33) **EXPLAIN WHAT TEST PLAN IS ? WHAT IS THE INFORMATION THAT SHOULD BE COVERED ?**

A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

It includes key components like test objectives, test environments, test cases, and schedules. The purpose of a test plan is to ensure systematic and effective testing, identify defects, and improve software quality. Benefits include minimizing risks, enhancing product reliability, and meeting customer expectations.

34) **WHAT IS PRIORITY ?**

Priority is defined as a parameter that decides the order in which a defect should be fixed. Defects having a higher priority should be fixed first.

- Defects/ bugs that leave the software unstable and unusable are given higher priority over the defects that cause a small functionality of the software to fail.
- It refers to how quickly the defect should be rectified.

35) **WHAT IS SEVERITY ?**

Severity is defined as the extent to which a particular defect can create an impact on the software. Severity is a parameter to denote the implication and the impact of the defect on the functionality of the software.

- A higher effect of the bug on system functionality will lead to a higher severity level.
- A QA engineer determines the severity level of a bug.

36) **ADVANTAGE OF BUGZILA.**

Bugzilla is an open source bug-tracking system. It was developed by Mozilla Foundation in 1998. It was written in Perl programming language. Basically, it is used as a bug-reporting tool in the market. Companies that use Bugzilla are Frido, My Stack, Adroiti Technologies, CloudByte and many more.

➢ **Advantages of Bugzilla**

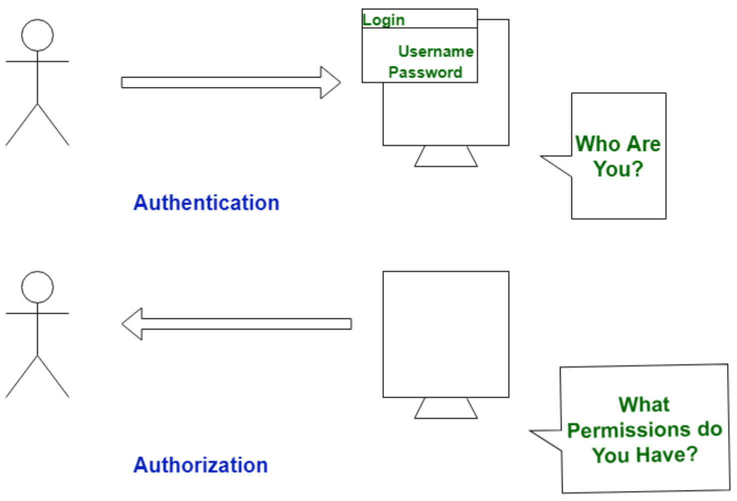
- **Deadlines:** To fix the bugs, deadlines can be established.
- **Types:** It reports in a variety of formats and types.
- **Request System:** You can use the 'request system' provided by Bugzilla to ask other users to evaluate codes, provide information and other things.
- **Flexible:** Bugzilla is quite flexible, so you can modify it to fit your unique process and requirements.
- **Bug tracking tool:** Bugzilla is extremely good at monitoring and handling bugs and issues.

37) **WHAT ARE THE DIFFERENT METHODOLOGIES IN AGILE DEVELOPMENT MODEL ?**

- **Models:** Crystal Agile Software Development Methodology places a strong emphasis on fostering effective communication and collaboration among team members, as well as taking into account the human elements that are crucial for a successful development process. This methodology is particularly beneficial for projects with a high degree of uncertainty, where requirements tend to change frequently.
- **Atern:** This methodology is tailored for projects with moderate to high uncertainty where requirements are prone to change frequently. Its clear-cut roles and responsibilities focus on delivering working software in short time frames. Governance practices set it apart and make it an effective approach for teams and projects.
- **Feature-driven development:** This approach is implemented by utilizing a series of techniques, like creating feature lists, conducting model evaluations, and implementing a design-by-feature method, to meet its goal. This methodology is particularly effective in ensuring that the end product is delivered on time and that it aligns with the requirements of the customer.
- **Scrum:** This methodology serves as a framework for tackling complex projects and ensuring their successful completion. It is led by a Scrum Master, who oversees the process, and a Product Owner, who establishes the priorities. The Development Team, accountable for delivering the software, is another key player.
- **Extreme programming (XP):** It uses specific practices like pair programming, continuous integration, and test-driven development to achieve these goals. Extreme programming is ideal for projects that have high levels of uncertainty and require frequent changes, as it allows for quick adaptation to new requirements and feedback.
- **Lean Development:** It is rooted in the principles of lean manufacturing and aims to streamline the process by identifying and removing unnecessary steps and activities. This is achieved through practices such as continuous improvement, visual management, and value stream mapping, which helps in identifying areas of improvement and implementing changes accordingly.

❖ **Unified Process:** Unified Process is a methodology that can be tailored to the specific needs of any given project. It combines elements of both waterfall and Agile methodologies, allowing for an iterative and incremental approach to development. This means that the UP is characterized by a series of iterations, each of which results in a working product increment, allowing for continuous improvement and the delivery of value to the customer.

38) **EXPLAIN THE DIFFERENCE BETWEEN AUTHORIZATION , AUTHENTICATION IN WEB TESTING. WHAT ARE THE COMMON PROBLEMS FACED IN WEB TESTING ?**



Authentication	Authorization
In the authentication process, the identity of users are checked for providing the access to the system.	While in authorization process, a the person's or user's authorities are checked for accessing the resources.
In the authentication process, users or persons are verified.	While in this process, users or persons are validated.
It is done before the authorization process.	While this process is done after the authentication process.
It needs usually the user's login details.	While it needs the user's privilege or security levels.
Authentication determines whether the person is user or not.	While it determines What permission does the user have?
Generally, transmit information through an ID Token.	Generally, transmit information through an Access Token.
The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process.	The OAuth 2.0 protocol governs the overall system of user authorization process.
Popular Authentication Techniques- Password-Based Authentication Passwordless Authentication 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) Single sign-on (SSO) Social authentication	Popular Authorization Techniques- Role-Based Access Controls (RBAC) JSON web token (JWT) Authorization SAML Authorization OpenID Authorization OAuth 2.0 Authorization
The authentication credentials can be changed in part as and when required by the user.	The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it.
The user authentication is visible at user end.	The user authorization is not visible at the user end.

The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc.	The user authorization is carried out through the access rights to resources by using roles that have been pre-defined.
Example: Employees in a company are required to authenticate through the network before accessing their company email.	Example: After an employee successfully authenticates, the system determines what information the employees are allowed to access.

Testing a web application can be a very hectic task. With the number of devices and browsers increasing day by day, apart from testing the functionality and performance, proper testing needs to be conducted to ensure that there is no impact on the overall user experience of the website. In this article, we shall discuss 16 challenges often faced by testers while testing a web application.

1. **Cross Browser Compatibility**

Earlier, when internet explorer was the only browser available, just unit testing would have done the job. But, currently, with hundreds of browsers along with their different versions available for desktop and mobile, cross-browser compatibility is a common issue. It is ideal for a tester to use a cloud testing platform like Lambdatest for testing whether the application is compatible across different browsers.

2. **Responsiveness**

The one thing to look out for while testing is whether the application fits properly in the device resolution. A tester must check if there are any horizontal scrolling, alignment or padding issues, and sizes of font and buttons in different devices.

It's of utmost importance that your images are responsive to different resolutions. In this case use LT Browser a dev-friendly browser for mobile view debugging on which you can see the mobile view of your website on Android and iOS resolutions.

3. **Cross-Device Compatibility**

Nowadays, people mostly use mobile devices to access websites. Although there are a limited number of devices in iOS, the count increases tenfold when it comes to Android. It is important for a tester to target the devices where the application is meant to run and start testing in each of them.

4. **Integration Testing**

The rating of an application depends on its usability as well as functionality. Integration testing is a must thing to carry out at the user's end to check whether the application is reliable, all the critical functionalities work properly as well as there is no significant impact on performance after merging new features.

5. **Security**

If the application has features like online transaction and payment gateways, testing should be executed to ensure that there are no chance of any fraudulent activities and local storage of payment-related data in the device.

6. **Performance Testing**

Often a web application gets too slow or crashes when the internet traffic increases all of a sudden. Performance testing should be carried out to ensure that there is no impact on the speed of performing an activity using the application.

7. **Application Getting Slow**

It does not matter what device is used to access the application, due to poor network coverage or low configuration of processor or physical memory an application may run slower or take an infinite time to load a page. Testing should be conducted to ensure that it is properly optimized to run properly under any condition.

8. Usability Testing

Interactive and dynamic web applications are always popular among users. Proper unit testing should be carried out across devices from the user's perspective to ensure there are no such issues that may impact the usability of the application.

9. Entry and Exit Points

There are stages when a user will need to navigate out from the application to a third-party website and redirect from another website or gateway to the application. It is a real challenge to test whether this feature works properly.

10. Checking the Standards and Compliance

W3C has stated several standards and guidelines that every web application must comply to. To ensure proper site ranking in the search engine index, the code should be tested properly to check whether the website follows those standards and guidelines.

11. Firewalls

Often a web application is blocked by certain firewalls or port. This may be because of the security certificate or something else. Testing should be conducted to ensure that it behaves properly across all firewalls.

12. Accessibility Testing

W3C has mentioned several guidelines stated in Section 508 and WCAG which requires a website to be accessible by all people, especially people with disabilities. Testing should be conducted to ensure that users with hearing or sight disabilities can access the website with the use of screen-reader and other devices.

13. Project Deadline

Testing is often not conducted properly when a project is coming nearer to the deadline. It should be planned beforehand to ensure that there is a proper time for testing the functionality, performance, and usability of the application before it is deployed in production.

14. User Experience

Users love an application not only based on its functionality. But also on how great it is to use. Testers must check the overall design and user experience of the application to ensure that the user gets attracted and engaged while using it.

15. Web Service Requests

All the latest web applications are integrated with web service layers like SOAP/JSON/XML, etc. The requests sent by those web services contain certain parameter values that need to be tested. This testing is particularly important if your web application has a Google Map API integrated with it.

16. User Input Validation

This is an important part often skipped during testing. Web applications may often feature forms to be filled up by users. Testing should be conducted to ensure that there are proper validation rules and in case of wrong input, the user is warned and further proceeding with the form submission is blocked until the input is corrected.

39) **WHEN TO USE USABILITY TESTING ?**

In an ideal world, the correct answer to "When should you run a usability test?" would simply be "early and often."

- **When It's Time to Evaluate and Iterate**

Broadly speaking, there are two kinds of user research: generative and evaluative.

Generative research helps you define problems and product ideas. This is the kind of research we're using at the very beginning of the design process.

Evaluative research provides you with feedback about how something performs, like your prototype. It's also useful for getting insight on an existing product that needs to be redesigned.

Once you've made some design decisions and created a wireframe or prototype, you've transitioned to an evaluative phase.

At this point, we like to develop low-fidelity wireframes and conduct a range of behavioral research to get feedback on the design. This helps us rapidly iterate without having to change high-fidelity UI elements.

Next, we develop a clickable prototype, which is when we truly see how well our design solves the problem we set out to address.

Every test plan is different, but we're generally looking to learn the following things from usability tests at this stage:

1. How many people completed a pre-defined task
2. If they completed a task, how quickly were they able to do so?
3. Was the process intuitive?

Taken as a whole, this kind of data can validate or invalidate our design, and give us a clear understanding of where we should continue to iterate.

- **After Launch**

In a philosophical sense, is a design ever really "done"?

There's always something that can be improved, and if your client wants to continue optimizing their design — or if they want to start developing new features — it's important to continue running usability tests.

During this part of the research, we use daily studies to supplement the data we get from ongoing behavioral research.

- **In High-Risk, Low-Certainty Situations**

When there's uncertainty in the design process, organizing a usability test can almost always provide some answers.

Uncertainty could take the form of a disagreement between team members. It could look like an absurd request from an executive that you're 99% certain will fail.

Either way, well-structured user research can help you make more informed decisions.

40) **WHAT IS THE PROCEDURE FOR GUI TESTING ?**

Step 1: Understand the Requirements

The first step is to fully understand what the software application is supposed to do. What are its intended functionalities? How should it behave under different conditions? By having a clear picture of the application's purpose, you can form a strong foundation for your GUI Testing process.

Step 2: Define the Test Cases

Based on the requirements, define a set of test cases to cover all aspects of the application's GUI. This should include tests for every button, menu, text field, and all other visual components. It's also important to plan tests for the software's various functionalities and possible user interactions.

Step 3: Set Up the Test Environment

Prepare the environment in which the tests will be conducted. This includes the hardware, software, and network configurations that mirror the end user's setup as closely as possible. Make sure to also plan for testing across different platforms and devices to ensure compatibility.

Step 4: Execute the Tests

Now, it's time to run the tests. This can be done manually, with a human tester going through each test case, or automatically, using specialized testing software. Often, a combination of both methods is used for comprehensive testing. As each test is run, carefully record the results for later analysis.

Step 5: Report and Analyze

Once all the tests have been run, compile the results into a report. This should detail which tests passed, which failed, and any anomalies or issues that were encountered. Analyze these results to identify any trends or recurring issues that might indicate a deeper problem.

Step 6: Make Necessary Changes

Based on your analysis, make any necessary changes to the software application. This might involve fixing bugs, tweaking the interface, or improving performance. Once the changes are made, repeat the testing process to ensure the issues have been resolved.

Step 7: Review and Improvement

Finally, review the entire testing process. What went well? What could be improved? By learning from each round of testing, you can continuously refine your GUI Testing process, making it more effective and efficient with each iteration.

41) BUG CATEGORIES ARE…

Bug categories in software development typically include:

1. Functional Bugs: These are issues where the software does not behave as intended or specified in terms of its functionality.

2. Performance Bugs: These bugs impact the performance of the software, such as slow response times, high resource usage, or inefficient algorithms.

3. Compatibility Bugs: These occur when the software behaves differently on different platforms, browsers, or environments, or when it fails to integrate properly with other software or systems.

4. Usability Bugs: These are related to the user experience, including issues with interface design, navigation, clarity of instructions, etc.

5. Security Bugs: These are vulnerabilities in the software that can be exploited to compromise its security, such as input validation errors, authentication issues, or insecure configurations.

6. *Regression Bugs: These occur when a feature that used to work correctly becomes broken after changes are made elsewhere in the codebase.

7. Interface Bugs: These involve problems with communication between different components of the software, such as APIs, libraries, or modules.

8. Documentation Bugs: These are discrepancies or errors in the documentation of the software, including missing or outdated information, unclear instructions, or incorrect examples.

9. Localization Bugs: These occur when the software does not support multiple languages or regions properly, resulting in mistranslations, formatting issues, or cultural insensitivity.

10. Hardware Bugs: While less common in software development, these bugs relate to issues with the interaction between the software and the underlying hardware, such as compatibility problems or driver issues.

42) **TEST CASE OF INSTAGRAM LOG IN PAGE :-**
https://github.com/pankti2712/software-testing/blob/main/TEST%20CASES/INSTAGRAM%20LOG%20IN%20PAGE.xlsx

43) **TEST CASE OF INSTAGRAM WEB LOG IN PAGE :-**

https://github.com/pankti2712/software-testing/blob/main/TEST%20CASES/INSTAGRAM%20WEB%20LOGIN%20PAGE.xlsx

44) **TEST CASE OF FACE BOOK LOG IN PAGE :-**

https://github.com/pankti2712/software-testing/blob/main/TEST%20CASES/FACEBOOK%20LOGIN%20PAGE.xlsx

45) **TEST CASE OF ART OF TESTING :-**

https://github.com/pankti2712/software-testing/blob/main/TEST%20CASES/ART%20OF%20TESTING.xlsx

46) **TEST CASE OF WHATS APP WEB :-**

https://github.com/pankti2712/software-testing/blob/main/TEST%20CASES/WHATS%20APP%20WEB.xlsx

47) **HLR OF WHATS APP WEB :-**

https://github.com/pankti2712/software-testing/blob/main/HLR/HLR%20OF%20WHATS%20APP%20WEB.xlsx

48) **HLR OF ART OF TESTING :-**

https://github.com/pankti2712/software-testing/blob/main/HLR/HLR%20of%20ART%20OF%20TESTING.xlsx

49) HLR OF INSTAGRAM LOG IN PAGE :-

https://github.com/pankti2712/software-testing/blob/main/HLR/HLR%20of%20Instagram%20Login%20Page.xlsx

50) HLR OF FACEBOOK LOG IN PAGE :-

https://github.com/pankti2712/software-testing/blob/main/HLR/HLR%20of%20Facebook%20Login%20Page.xlsx

51) HLR OF INSTAGRAM WEB LOG IN PAGE :-

https://github.com/pankti2712/software-testing/blob/main/HLR/HLR%20of%20Instagram%20Web%20Login%20Page.xlsx

52) Write a scenario of only Whatsapp chat messages .

- Verify that users can send text messages to individual contacts.
- Verify that users can send text messages to multiple contacts in a group chat.
- Verify that sent text messages are delivered promptly to the recipient(s).
- Verify that users can send emojis and emoticons in text messages.
- Verify that users can send multimedia messages such as photos and videos to individual contacts.
- Verify that users can send multimedia messages to multiple contacts in a group chat.
- Verify that sent multimedia messages are delivered promptly and display correctly to the recipient(s).
- Verify that users can send voice messages to individual contacts.
- Verify that users can send voice messages to multiple contacts in a group chat.
- Verify that sent voice messages are delivered promptly and play correctly for the recipient(s).
- Verify that users can send documents and files (e.g., PDFs, Word documents) to individual contacts.
- Verify that users can send documents and files to multiple contacts in a group chat.
- Verify that sent documents and files are delivered promptly and can be downloaded by the recipient(s).
- Verify that users can forward received messages to other contacts or groups.
- Verify that users can reply to specific messages within a chat thread.
- Verify that users can delete messages they have sent within a certain timeframe.
- Verify that users can edit messages they have sent within a certain timeframe.
- Verify that users receive notifications for new incoming messages.
- Verify that users can mute notifications for specific chats or contacts.
- Verify that users can mark messages as unread to remind themselves to respond later.
- Verify that users can search for specific messages within a chat or group chat.
- Verify that users can star important messages to access them quickly later.
- Verify that users can archive chats to declutter their chat list.
- Verify that users can backup chat messages and media to cloud storage services.
- Verify that users can restore chat messages and media from cloud backups when switching devices.
- Verify that users can block contacts to prevent them from sending messages.
- Verify that users can report inappropriate messages or contacts to WhatsApp for review.
- Verify that users can set custom chat wallpapers and themes for individual chats or group chats.

- Verify that users can set custom notification sounds for individual chats or group chats.
- Verify that users can view read receipts (blue ticks) to confirm when messages have been read by the recipient(s).
- Verify that users can view delivery receipts (single tick) to confirm when messages have been successfully delivered to the recipient(s).
- Verify that users can see the online status of contacts to know when they are active on WhatsApp.
- Verify that users can see typing indicators to know when contacts are composing messages.
- Verify that users can access chat settings to customize privacy, security, and notification preferences.
- Verify that users can access chat settings to export chat history or delete chat data when needed.

53) Write a Scenario of Pen .

- Verify that the pen writes smoothly on various types of paper surfaces.
- Verify that the pen's ink flow is consistent without skipping or blotting.
- Verify that the pen's ink dries quickly to prevent smudging or smearing.
- Verify that the pen's grip is comfortable and ergonomic for extended writing sessions.
- Verify that the pen's barrel is durable and resistant to cracks or breakage.
- Verify that the pen's cap (if applicable) securely covers the tip to prevent ink leakage.
- Verify that the pen's retractable mechanism (if applicable) functions smoothly and securely.
- Verify that the pen's clip securely attaches to pockets, notebooks, or bags without coming loose.
- Verify that the pen's ink color matches the indicated color on the pen's packaging or labeling.
- Verify that the pen's ink is waterproof and fade-resistant when exposed to water or sunlight.
- Verify that the pen's tip size (e.g., fine, medium, bold) matches the indicated size on the pen's packaging or labeling.
- Verify that the pen's ink cartridge is easy to replace when the ink runs out.
- Verify that the pen's ink cartridge has a sufficient ink capacity for prolonged use.
- Verify that the pen's design aesthetic is appealing and suitable for its intended use or target audience.
- Verify that the pen's manufacturer provides information on ink refill options and compatibility with other ink cartridges.
- Verify that the pen's packaging is secure and protects the pen from damage during shipping and handling.
- Verify that the pen's price is competitive compared to similar pens in the market.
- Verify that the pen's manufacturer provides responsive customer support and assistance with any inquiries or issues.
- Verify that the pen's ink formula is non-toxic and safe for use by adults and children.
- Verify that the pen's ink does not bleed through or feather on thin paper.
- Verify that the pen's ink remains consistent in color and intensity throughout its lifespan.
- Verify that the pen's ink does not clog or dry out when stored for extended periods.
- Verify that the pen's manufacturer provides information on sustainable and eco-friendly manufacturing practices.

54) Write a Scenario of Pen Stand .

- Verify that the pen stand is stable and does not tip over easily when loaded with pens.
- Verify that the pen stand has sufficient capacity to hold the desired number of pens without overcrowding.

- Verify that the pen stand's compartments or slots are appropriately sized to accommodate different types and sizes of pens.
- Verify that the pen stand's material (e.g., plastic, metal, wood) is durable and resistant to damage from everyday use.
- Verify that the pen stand's design allows for easy access to pens and prevents them from becoming tangled or jammed.
- Verify that the pen stand's finish (e.g., paint, coating) is smooth and free from defects or rough edges.
- Verify that the pen stand's weight distribution is balanced to prevent it from wobbling or sliding on surfaces.
- Verify that the pen stand's base is non-slip and provides stability on various surfaces such as desks, tables, or countertops.
- Verify that the pen stand's construction is environmentally friendly and complies with relevant regulations and standards.
- Verify that the pen stand's color and design complement the aesthetics of the surrounding environment.
- Verify that the pen stand's assembly process (if required) is straightforward and does not require specialized tools or expertise.
- Verify that the pen stand's compartments are deep enough to prevent pens from falling out easily.
- Verify that the pen stand's overall size and shape fit comfortably on desks or workstations without taking up excessive space.
- Verify that the pen stand's compartments are evenly spaced to prevent overcrowding and facilitate organization.
- Verify that the pen stand's material is resistant to fading or discoloration over time, especially when exposed to sunlight or other environmental factors.
- Verify that the pen stand's edges and corners are smooth and rounded to prevent injury or damage to pens.
- Verify that the pen stand's design includes additional features such as storage compartments for other office supplies (e.g., paper clips, sticky notes).
- Verify that the pen stand's packaging protects it from damage during shipping and handling.
- Verify that the pen stand's price is competitive compared to similar products in the market.
- Verify that the pen stand's manufacturer provides responsive customer support and assistance with any inquiries or issues.

55) Write a Scenario of Door .

- Verify that the door opens and closes smoothly without sticking or jamming.
- Verify that the door latch securely engages when closed, preventing the door from opening unintentionally.
- Verify that the door handle or knob is functional and operates properly to open and close the door.
- Verify that the door hinges are secure and properly lubricated, allowing the door to swing freely.
- Verify that the door frame is structurally sound and provides a stable mounting surface for the door.
- Verify that the door's locking mechanism, if present, functions correctly to secure the door when locked.
- Verify that the door's lock cylinder (if applicable) accepts and turns keys smoothly without resistance.

- Verify that the door's deadbolt (if applicable) extends and retracts smoothly when locked and unlocked.
- Verify that the door's weatherstripping (if installed) effectively seals gaps around the door perimeter to prevent drafts and air leakage.
- Verify that the door's threshold (if present) is properly aligned and provides a smooth transition between the interior and exterior surfaces.
- Verify that the door's sound insulation properties meet or exceed specified standards, reducing noise transmission between rooms or outdoor environments.
- Verify that the door's fire rating (if applicable) meets local building code requirements for fire resistance.
- Verify that the door's material (e.g., wood, metal, fiberglass) is durable and resistant to damage from weather, impacts, and wear over time.
- Verify that the door's finish (e.g., paint, stain, varnish) is applied evenly and provides protection against moisture and UV exposure.
- Verify that the door's glass panels (if present) are shatter-resistant and meet safety standards for impact resistance.
- Verify that the door's hinges and hardware (e.g., screws, bolts, fasteners) are securely tightened and properly installed.
- Verify that the door's swing direction (inward or outward) is appropriate for its location and does not obstruct traffic flow or access.
- Verify that the door's width and height dimensions match the intended opening and allow for sufficient clearance when opening and closing.
- Verify that the door's operation is consistent across different temperatures and weather conditions.
- Verify that the door's design aesthetic complements the architectural style of the building or room.
- Verify that the door's installation meets industry best practices and manufacturer recommendations for proper alignment and functionality.
- Verify that the door's warranty coverage is accurately described and provides adequate protection against defects or damage.

56) Write a Scenario of ATM .

- Verify that the ATM machine is powered on and operational.
- Verify that the ATM's display screen is functional and displays relevant information such as available services and instructions.
- Verify that users can insert their bank card into the ATM's card reader slot.
- Verify that the ATM's card reader reads bank cards accurately and without errors.
- Verify that the ATM prompts users to enter their PIN code securely.
- Verify that the ATM's PIN pad is functional and responsive to user input.
- Verify that users can select their desired transaction type (e.g., withdrawal, deposit, balance inquiry) from the ATM's menu.
- Verify that the ATM accurately processes withdrawal transactions and dispenses the correct amount of cash.
- Verify that the ATM's cash dispenser mechanism operates smoothly without jams or errors.
- Verify that the ATM's cash bin is adequately stocked with bills of various denominations.
- Verify that users receive a printed receipt for their transaction if requested.
- Verify that the ATM's receipt printer is operational and prints receipts clearly.

- Verify that the ATM's deposit function accepts cash and checks securely.
- Verify that the ATM's deposit slot is wide enough to accommodate envelopes containing deposits.
- Verify that users can view their account balance accurately through the ATM's interface.
- Verify that the ATM's balance inquiry function displays account balances for different linked accounts if applicable.
- Verify that the ATM's menu options are displayed in a clear and intuitive manner.
- Verify that the ATM's language options are accessible and accurately translated.
- Verify that the ATM's audio instructions (if available) are clear and easy to understand.
- Verify that the ATM's transaction processing time is within acceptable limits.
- Verify that the ATM's receipt includes transaction details such as date, time, amount, and account balance.
- Verify that the ATM's card return slot returns the user's bank card securely after the transaction is completed.
- Verify that the ATM's transaction cancellation function allows users to cancel transactions safely if needed.
- Verify that the ATM's security features, such as surveillance cameras and anti-skimming technology, are operational.
- Verify that the ATM's cash replenishment process is scheduled regularly to ensure availability of funds.
- Verify that the ATM's network connectivity is stable and reliable for processing transactions.
- Verify that the ATM's user interface is accessible to users with disabilities, including visually impaired users.
- Verify that the ATM's customer support contact information is displayed prominently for assistance with issues or inquiries.
- Verify that the ATM's customer support contact information is displayed prominently for assistance with issues or inquiries.

57) Write a scenario of Microwave Owen .

- Verify that the microwave oven is plugged in and receiving power.
- Verify that the microwave oven's control panel is functional and responsive to user input.
- Verify that the microwave oven's door opens and closes smoothly without sticking.
- Verify that the microwave oven's interior is clean and free from debris or residue.
- Verify that the microwave oven's turntable (if applicable) rotates smoothly when in operation.
- Verify that the microwave oven's heating elements produce heat as expected.
- Verify that the microwave oven's cooking presets (e.g., popcorn, defrost) function correctly and produce desired results.
- Verify that the microwave oven's timer accurately counts down cooking time and alerts when cooking is complete.
- Verify that the microwave oven's power levels can be adjusted and result in varying cooking intensities.
- Verify that the microwave oven's door latch securely locks during operation to prevent accidental opening.
- Verify that the microwave oven's interior light illuminates when the door is opened for easy visibility.
- Verify that the microwave oven's safety features, such as child lock, are functional and effective.
- Verify that the microwave oven's venting system efficiently removes steam and odors during cooking.

- Verify that the microwave oven's exterior remains cool to the touch during operation to prevent burns.
- Verify that the microwave oven's cooking chamber is adequately insulated to prevent heat loss.
- Verify that the microwave oven's door seal is intact and prevents leakage of microwaves.
- Verify that the microwave oven's door glass is transparent and free from cracks or damage.
- Verify that the microwave oven's cooking performance is consistent across different food items and quantities.
- Verify that the microwave oven's user manual provides clear instructions for operation and maintenance.
- Verify that the microwave oven complies with safety standards and regulations for household appliances.
- Verify that the microwave oven's warranty coverage is accurately described and accessible to users.
- Verify that the microwave oven's overall build quality and durability meet expectations for long-term use.
- Verify that the microwave oven's cooking chamber size accommodates standard cookware and food containers effectively.
- Verify that the microwave oven's power cord and plug are undamaged and meet electrical safety standards cookware and food containers effectively.
- Verify that the microwave oven's power cord and plug are undamaged and meet electrical safety standards.

58) Write a scenario of Coffee vending Machine .

- Verify that the vending machine is powered on and operational.
- Verify that the vending machine is properly stocked with coffee beans, milk, water, and other necessary ingredients.
- Verify that the vending machine's display screen is functional and displays relevant information such as available drinks and prices.
- Verify that users can select their desired coffee type or customization options from the vending machine's menu.
- Verify that users can insert payment (e.g., coins, bills, card) into the vending machine and that payment processing is reliable.
- Verify that the vending machine dispenses the selected coffee drink promptly and accurately.
- Verify that the coffee temperature is at the desired level for consumption.
- Verify that the coffee quality meets expectations in terms of taste, aroma, and consistency.
- Verify that the vending machine's dispensing mechanism operates smoothly without jams or errors.
- Verify that the vending machine provides options for cup size and quantity (e.g., small, medium, large).
- Verify that users can add optional flavorings or condiments to their coffee (e.g., sugar, cream).
- Verify that the vending machine offers a selection of hot and cold coffee beverages, if applicable.
- Verify that the vending machine's water temperature for hot drinks is within the acceptable range.
- Verify that the vending machine's milk frothing mechanism (if applicable) produces frothy milk for drinks like cappuccinos and lattes.
- Verify that the vending machine's cleaning and maintenance schedule is followed regularly to ensure hygiene and functionality.
- Verify that the vending machine's user interface is intuitive and easy to navigate.

- Verify that the vending machine's buttons or touch screen respond accurately to user input.
- Verify that the vending machine's cup dispensing mechanism provides cups without defects or contamination.
- Verify that the vending machine's waste disposal system effectively collects used cups and waste materials.
- Verify that the vending machine's internal sensors detect and alert maintenance staff to issues such as low ingredient levels or malfunctions.
- Verify that the vending machine's power-saving features are enabled when not in use to conserve energy.
- Verify that the vending machine's overall performance meets or exceeds specified standards for reliability and customer satisfaction.
- Verify that the vending machine's overall performance meets or exceeds specified standards for reliability and customer satisfaction.

59) Write a scenario of chair .

- Verify that the chair's dimensions match the specifications provided by the manufacturer.
- Verify that the chair's construction materials meet quality standards and are durable.
- Verify that the chair's design aligns with ergonomic principles for comfort and support.
- Verify that the chair's weight capacity matches the intended usage and is indicated accurately.
- Verify that the chair's assembly instructions are clear and easy to follow.
- Verify that all necessary assembly hardware and tools are included with the chair.
- Verify that the chair's frame is sturdy and stable when assembled.
- Verify that the chair's legs or base have appropriate non-slip features to prevent sliding.
- Verify that the chair's seat and backrest are comfortable and provide adequate support.
- Verify that the chair's upholstery, if applicable, is of high quality and free from defects.
- Verify that the chair's armrests, if present, are adjustable and provide comfortable support.
- Verify that the chair's height adjustment mechanism, if applicable, works smoothly and securely locks into place.
- Verify that the chair's swivel function, if applicable, operates smoothly and without resistance.
- Verify that the chair's recline mechanism, if applicable, functions as intended and locks securely in position.
- Verify that the chair's casters or wheels, if applicable, roll smoothly on different floor surfaces without leaving marks.
- Verify that the chair's finish (e.g., paint, varnish) is uniform and free from scratches or blemishes.
- Verify that the chair's packaging protects it from damage during shipping and handling.
- Verify that the chair complies with relevant safety standards and regulations.
- Verify that the chair's warranty information is provided and accurately reflects the coverage offered.
- Verify that the chair's price is competitive compared to similar products in the market.
- Verify that the chair's design aesthetic complements various styles of decor.
- Verify that the chair's customer reviews and ratings are positive and reflect satisfaction with the product.
- Verify that the chair's shipping and delivery process is efficient and punctual.
- Verify that the chair's manufacturer or retailer provides responsive customer support for inquiries or issues.

60) To Create Scenario (Positive & Negative) .

- **Gmail (receiving mail) :-**

  Verify that users can search for products using keywords.
  - Verify that search results display relevant products based on the search query.
  - Verify that users can filter search results by various criteria such as price, brand, ratings, and availability.
  - Verify that product details pages provide comprehensive information including price, description, specifications, images, and availability.
  - Verify that users can add products to their shopping cart.
  - Verify that the shopping cart accurately reflects the added products, including quantity and price.
  - Verify that users can edit the quantity or remove products from the shopping cart.
  - Verify that users can proceed to checkout from the shopping cart page.
  - Verify that users can choose their preferred payment method (e.g., credit/debit card, net banking, cash on delivery).
  - Verify that users can provide shipping address details during checkout.
  - Verify that users can apply discounts or promotional codes during checkout (if applicable).
  - Verify that users receive an order summary before confirming the purchase.
  - Verify that users can place the order successfully.
  - Verify that users receive an order confirmation message with order details.
  - Verify that users receive updates on their order status (e.g., processing, shipped, delivered).
  - Verify that users can track their orders from their Flipkart account.
  - Verify that users can view their order history and previous purchases.
  - Verify that users receive notifications for any changes or updates related to their orders.
  - Verify that users can contact customer support for assistance with their orders.
  - Verify that the checkout process is secure and user data is protected.
  - Verify that users can provide feedback or ratings for products they have purchased.
  - Verify that users can initiate returns or refunds for products if necessary.
  - Verify that users can share their shopping experience on social media platforms.
  - Verify that users can access help and support resources for any issues encountered during the shopping process.

61) shopping to buy product (flipkart) .

- Verify that users can search for products using keywords.
- Verify that search results display relevant products based on the search query.
- Verify that users can filter search results by various criteria such as price, brand, ratings, and availability.
- Verify that product details pages provide comprehensive information including price, description, specifications, images, and availability.
- Verify that users can add products to their shopping cart.
- Verify that the shopping cart accurately reflects the added products, including quantity and price.
- Verify that users can edit the quantity or remove products from the shopping cart.
- Verify that users can proceed to checkout from the shopping cart page.
- Verify that users can choose their preferred payment method (e.g., credit/debit card, net banking, cash on delivery).
- Verify that users can provide shipping address details during checkout.
- Verify that users can apply discounts or promotional codes during checkout (if applicable).

- Verify that users receive an order summary before confirming the purchase.
- Verify that users can place the order successfully.
- Verify that users receive an order confirmation message with order details.
- Verify that users receive updates on their order status (e.g., processing, shipped, delivered).
- Verify that users can track their orders from their Flipkart account.
- Verify that users can view their order history and previous purchases.
- Verify that users receive notifications for any changes or updates related to their orders.
- Verify that users can contact customer support for assistance with their orders.
- Verify that the checkout process is secure and user data is protected.
- Verify that users can provide feedback or ratings for products they have purchased.
- Verify that users can initiate returns or refunds for products if necessary.
- Verify that users can share their shopping experience on social media platforms.
- Verify that users can access help and support resources for any issues encountered during the shopping process.

62) Write a Scenario of Wrist Watch .

- Verify that the watch accurately displays the current time.
- Verify that the watch's date function displays the correct date.
- Verify that the watch's alarm function works as intended.
- Verify that the watch's stopwatch function accurately measures time.
- Verify that the watch's timer function counts down accurately.
- Verify that the watch's backlight illuminates the display effectively in low-light conditions.
- Verify that the watch's buttons (e.g., for adjusting settings) are responsive and functional.
- Verify that the watch's straps or bands are securely attached and comfortable to wear.
- Verify that the watch's water resistance matches the stated specifications.
- Verify that the watch's battery life meets the manufacturer's claims.
- Verify that the watch's timekeeping remains accurate over an extended period.
- Verify that the watch's design and aesthetics meet user expectations.
- Verify that the watch's materials are durable and resistant to wear and tear.
- Verify that the watch's display is easy to read in various lighting conditions.
- Verify that the watch's settings (e.g., time zone adjustment) can be configured easily.
- Verify that the watch's crown (if applicable) functions correctly for setting time and date.
- Verify that the watch's notifications (if applicable) are delivered accurately and promptly.
- Verify that the watch's straps or bands are interchangeable (if advertised as such).
- Verify that the watch's clasp or buckle securely fastens the straps or bands.
- Verify that the watch's scratch-resistant coating (if applicable) effectively protects the display.
- Verify that the watch's synchronization with external time sources (e.g., atomic clock) is accurate.
- Verify that the watch's size and weight are comfortable for daily wear.
- Verify that the watch's buttons are resistant to accidental presses.
- Verify that the watch's manufacturer provides reliable customer support and warranty services.

63) Write a Scenario of Lift (Elevator) .

- Verify that the elevator doors open and close smoothly.
- Verify that the elevator responds promptly to user input (pressing floor buttons).
- Verify that the elevator stops accurately at selected floors.
- Verify that the elevator can handle multiple simultaneous requests from different floors.

- Verify that emergency buttons and alarms function properly.
- Verify that the elevator can accommodate its maximum weight capacity.
- Verify that the elevator provides audible and visual signals for floor arrival.
- Verify that the elevator maintains proper alignment with each floor level.
- Verify that the elevator operates smoothly without sudden jerks or stops.
- Verify that the elevator doors remain open for an appropriate duration.
- Verify that the elevator's emergency backup power system functions correctly.
- Verify that the elevator's safety features, such as sensors, are operational.
- Verify that the elevator's emergency evacuation procedures are clear and accessible.
- Verify that the elevator's display panel accurately indicates the current floor and direction of travel.
- Verify that the elevator's cabin is clean and well-maintained.
- Verify that the elevator's controls (buttons, touch panels) are responsive and functional.
- Verify that the elevator complies with safety regulations and standards.
- Verify that the elevator's speed is consistent and appropriate for the building's height.
- Verify that the elevator's doors do not close when obstructed.
- Verify that the elevator's ventilation system provides adequate airflow.
- Verify that the elevator's lighting is sufficient and properly distributed.
- Verify that the elevator's emergency communication system (intercom) works reliably.
- Verify that the elevator's maintenance records are up-to-date and accessible.
- Verify that the elevator's operation is quiet and minimally disruptive to occupants.

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

- Verify that users can create a new group.
- Verify that users can add contacts to the group.
- Verify that users can remove contacts from the group.
- Verify that group admins can manage group settings.
- Verify that group members can send messages within the group.
- Verify that group members receive notifications for group activities.
- Verify that group chat history is accessible to all members.
- Verify that media files (images, videos, documents) can be shared within the group.
- Verify that group members can leave the group voluntarily.
- Verify that group admins can remove members from the group.
- Verify that group settings, such as group description and privacy settings, are configurable.
- Verify that group invites can be sent to new members.
- Verify that group members can search for messages within the group chat.
- Verify that group calls can be initiated within the group (if applicable).
- Verify that group admins can assign other members as admins.
- Verify that group members can react to messages using emojis.
- Verify that group members can mention other members using '@' symbol.
- Verify that group members can change their nickname within the group.
- Verify that group members can view group information, including member list and group photo.
- Verify that group announcements or pinned messages are visible to all members.
- Verify that group settings can be customized, such as message notifications and media visibility.
- Verify that group members can report inappropriate content or behavior to admins.
- Verify that group data is backed up securely and can be restored if needed.

65) Write a Scenario of Whatsapp payment .

- Verify that registration process is smooth
- Verify that sending payments is reliable
- Verify that receiving payments is seamless
- Verify that payment notifications are received promptly
- Verify that payment details are accurate
- Verify that security features are effective
- Verify that transaction limits are enforce
- Verify that transaction history is accessible
- Verify that receipts can be generated
- Verify that error messages are informative
- Verify that customer support is
- Verify that payments work across platforms
- Verify that compliance with regulations is maintained
- Verify that performance is satisfactory under load
- Verify that accessibility features are functional