

Assignment

## Module-2(manual testing)

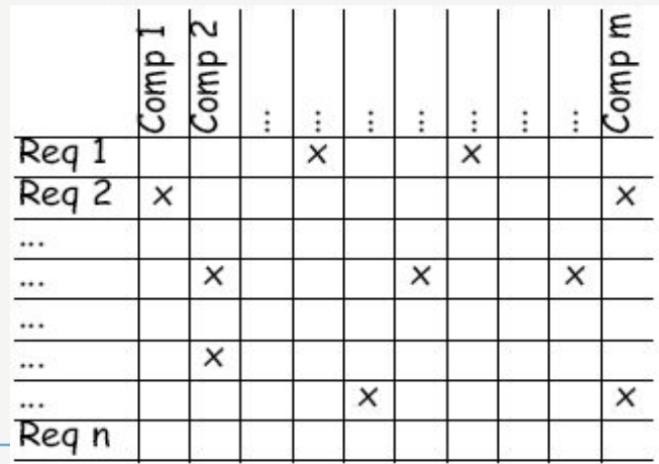
**Madhad krishna**

# 1. What is Exploratory Testing?

* **Ans**: We have to explore the application to understand the requirement completely, then only we will start testing.
* In simple words, to understand the requirement completely, first we have to explore the application, hence it is called as exploratory. Test engineer will do testing, when there is no requirement at all or the requirements are missing.
* Though the current trend in testing is to push for automation, exploratory testing is a new way of thinking. Automation has its limits.
* Is not random testing but it is adhoc testing with purpose of find bugs Is structured and rigorous Is cognitively (thinking) structured as compared to procedural structure of scripted testing.

# 2. What is traceability matrix?

* **Ans**: To protect against changes, you should be able to trace back from every system component to the original requirement that caused its presence.
* A software process should help you keeping the virtual table up-to- date.
* A software process should help you keeping the virtual table up-to-date
* Simple technique may be quite valuable (naming convention)



* Forward Traceability – Mapping of Requirements to Test cases
* Backward Traceability – Mapping of Test Cases to Requirements
* Bi-Directional Traceability - A Good Traceability matrix is the References from test

# 3. What is Boundary value testing?

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

# 

# What is Equivalence partitioning testing?

* **Ans**: Aim is to treat groups of inputs as equivalent and to select one representative input to test them all EP can be used for all Levels of Testing
* Equivalence partitioning is the process of defining the optimum number of tests by:
* Reviewing documents such as the Functional Design Specification and Detailed Design Specification, and identifying each input condition within a function,
* Selecting input data that is representative of all other data that would likely invoke the same process for that condition.
* The numbers fall into a partition where each would have the same, or equivalent, result i.e., an Equivalence Partition (EP) or Equivalence Class
* The Valid partition is bounded by the values 1 and 100
* Plus, there are 2 Invalid partitions

# what is integration testing?

* **Ans**: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.
* Integration Testing is a level of the software testing process where individual units are combined and tested as a group.
* Components may be code modules, operating systems, hardware and even complete systems
* There are 2 levels of Integration Testing.
* 1. Component Integration Testing
* 2. System Integration Testing

# What determines the level of risk?

# **Ans:** Determining the level of risk usually involves trying to assess not only the likelihood of an identified risk from actually occurring, but also the potential magnitude the consequences this risk could have on an organisation and its stakeholder, should it occur.

* As Risk is determined by a combination of Probability and Severity, the main area of the Matrix reveals the Risk Levels. The levels are Low, Medium, High, and Extremely High. To have a low level of risk, we must have a somewhat limited probability and level of severity

# what is Alpha testing?

* + **Ans**: It is always performed by the developers at the software development site.
  + Sometimes it is also performed by Independent Testing Team
  + t is always performed in Virtual Environment
  + It is the form of Acceptance Testing.
  + Alpha Testing is performed and carried out at the developing organization's location with the involvement of developers.
  + It comes under the category of both White Box Testing and Black Box Testing.
  + It is conducted for the software application and project

# What is Deta testing?

* + **Ans**: It is always performed by the customers at their own site.
  + It is not performed by Independent Testing Team.
  + Beta Testing is always open to the market and public.
  + It is performed in Real Time Environment
  + It is always performed outside the organization.
  + It is only a kind of Black Box Testing.
  + It is also the form of Acceptance Testing.
  + Beta Testing is always performed at the time when software product and project are marketed.
  + Beta testing can be considered “pre-release” testing.
  + Pilot Testing is testing to product on real world as well as collect data on the use of product in the classroom.

# What is component testing?

* + **Ans:** Component testing is a form of closed-box testing, meaning that the evaluates the behavior of the program without considering the details of the underlying code.
  + Component (Unit) – A minimal software item that can be tested in isolation. It means “A unit is the smallest testable part of software.”
  + Component Testing – The testing of individual software components.
  + Unit testing is performed by using the White Box Testing method.
  + Unit Testing is a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
  + Unit testing is the first level of testing and is performed prior to Integration Testing.
  + Sometimes known as Unit Testing, Module Testing or Program Testing
  + Component can be tested in isolation – stubs/drivers may be employed





# What is function system testing?

* + **Ans:** Functional Testing: Testing based on an analysis of the specification of the functionality of a component or system.
  + Function testing is a type of testing that verifies that each function of the software application operators in conformation with the requirement specification.
  + ‘Specification’ – E.g., Requirements specification, Use Cases, Functional specification or maybe undocumented.
  + ‘Function’ – what the system does
  + This testing mainly involves black box testing and it is not concerned about the source code of the application.
  + Each & every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
  + Functional Testing Examples:
* Web Based Testing:
* Are you able to login to a system after entering correct credentials?
* Desktop Based Testing:
* Verifies Installation testing, Check for broken lines,
* Mobile Based Testing:
* To validate whether the application works as per as requirement whenever the application starts/stops
* Game Based Testing:
* Takes more time to execute as testers look for game play issues,graphicsc issues, audio-visual issues, etc

# What is non-functional testing?

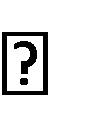
* Non-Functional Testing: Testing the attributes of a component or system that do not relate to functionality, e.g., reliability, efficiency, usability, interoperability, maintainability and portability.
* It is the testing of “how” the system works. Non-functional testing may be performed at all test levels.
* Non-functional testing includes, but is not limited to, performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing.
* To address this issue, performance testing is carried out to check & fine tune system response times. The goal of performance testing is to reduce response time to an acceptable level
* Hence load testing is carried out to check systems performance at different loads i.e., number of users accessing the system
* Non - Functional Testing Examples:
* Web Based Testing:
* In website number of user/customer will increase, how the website will handle to every customer/user.
* Desktop Based Testing:
* Guarantee that error messages are instructive and helpful for the client
* Mobile Based Testing:
* In mobile, automatically will switch off without any reason.
* Game Based Testing:
* Confirms workability and stability of the software.

# What is GUI testing?

* + Ans: Graphical User Interface (GUI) testing is the process of testing the system’s GUI of the System under Test. GUI testing involves checking the screens with the controls like menus, buttons, icons, and all types of bars – tool bar, menu bar, dialog boxes and windows etc.
  + GUI testing is a software testing type that checks the Graphical User Interface of the Software.
* Approach of GUI Testing:
  + 1. manual based testing
  + 2. record and replay
  + 3. model-based testing
* GUI Testing Examples:
* Web Based Testing & Desktop Based Testing:
* Enough space should be provided between field labels, columns, rows, error messages, etc.
* Mobile Based Testing:
* If mobile is in every orientation mode so display image, video properly
* Game Based Testing:
* Can be also conducted in advance of designing page layouts or navigation menus

# What is Adhoc testing?

* + Ans: Testing the application randomly, without looking into the
  + requirement is called adhoc testing.
  + also called as negative testing.
  + For this reason, it is also known as Random testing or Monkey testing.
  + Adhoc testing is an informal testing type with an aim to break the system.
  + Main aim of this testing is to find defects by random checking.
  + Adhoc testing can be achieved with the testing technique called Error Guessing.
  + The Error guessing is a technique where the experienced and good testers are encouraged to think of situations in which the software may not be able to cope.
  + Some people seem to be naturally good at testing and others are good testers because they have a lot of experience either as a tester or working with a particular system and so are able to find out its weaknesses.
  + This is why an error guessing approach, used after more formal techniques
* Types of Adhoc Testing:
* There are different types of Adhoc testing:
  + 1. Buddy Testing

 2. Pair testing

* + 3. Monkey Testing

# What is load testing?

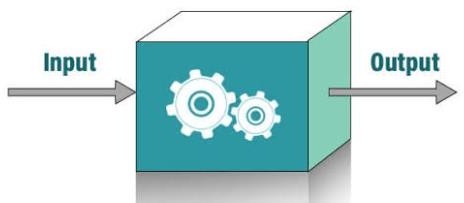
* + Ans: It's a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system’s response time degrades or fails.
  + Load testing is a kind of performance testing which determines a system’s performance under real-life load conditions. This testing helps determine how the application behaves when multiple users access it simultaneously.
  + This testing usually identifies –
  + The maximum operating capacity of an application
  + Determine whether current infrastructure is sufficient to run the application
  + It is a type of non-functional testing. Load testing is commonly used for the Client/Server, Web based applications – both Intranet and Internet.
  + Some extremely popular sites have suffered serious downtimes when they get massive traffic volumes. E-commerce websites invest heavily in advertising campaigns, but not in Load Testing to ensure optimal system performance, when that marketing brings in traffic.
* Strategies of Load Testing:
  + Manual Load Testing
  + In house (Organization) developed load testing tools
  + Open-source load testing tools Enterprise (Record and Play) load testing tools

# What is stress testing?

* + Ans: Stress testing - System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.
  + t even tests beyond the normal operating point and evaluates how the system works under those extreme conditions.
  + Stress Testing is done to make sure that the system would not crash under crunch situations.
  + Stress testing is also known as endurance testing.
  + Most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks.
  + Examples of stress conditions include:
  + Resource reduction such as a disk drive failure.
  + Application components fail to respond.
  + Need For Stress Testing:
  + During festival time, an online shopping site may witness a spike in traffic, or when it announces a sale.
  + When a blog is mentioned in a leading newspaper, it experiences a sudden surge in traffic.
* Types of Stress Testing:
  + Application Stress Testing
  + Transactional Stress Testing
  + Systemic Stress Testing
  + Exploratory Stress Testing:

# What is white box testing and list the types of white box testing?

* + Ans: White Box Testing: Testing based on an analysis of the internal structure of the component or system.
  + Structure-based testing technique is also known as ‘white-box’ or ‘glass-box’ testing technique because here the testers require knowledge of how the software is implemented, how it works
  + Testing based upon the structure of the code
  + White box testing is also called glass testing or open box testing.
  + The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately



### White Box Testing Examples:

* Web Based Testing:
* Analyze the logic by reading the code

### Types of white box:

* Statement coverage
* Decision coverage
* Condition coverage
* Test/Code Coverage

# What is black box testing and list the types of black box testing?

* + Ans: Black-box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.
  + Specification-based testing technique is also known as ‘black-box’
  + The testers have no knowledge of how the system or component is structured inside the box.
  + The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.
  + The tester is oblivious to the system architecture and does not have access to the source code.
  + Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
  + Black Box Testing Examples:
* Web Based Testing:
* Login by the user is must for accessing the sensitive information.
* Desktop Based Testing:
* Resolution change effect on the application
* Mobile Based Testing:
* In mobile, automatically will switch off without any reason.
* Game Based Testing:
* The game tester must know how to play the game, utilization of the gamepad, know the game flow and the rules.
  + Techniques of Black Box Testing:
  + There are four specification-based or black-box technique:
* Equivalence partitioning
* Boundary value analysis
* Decision tables State transition testing
* Use-case Testing
* Other Black Box Testing

# 18. Mention what are the categories of defects?

**Ans:**

There are 5 types of defects which are hear under:-

1. Data Quality/Database Defects: Deals with improper handling of data in the database.
2. Critical Functionality Defects: The occurrence of these bugs hampers the crucial functionality of the application.
3. Functionality Defects: These defects affect the functionality of the application.
4. Security Defects: Application security defects generally involve improper handling of data sent from the user to the application. These defects are the most severe and given highest priority for a fix.
5. User Interface Defects: As the name suggests, the bugs deal with problems related to UI are usually considered less severe

# Mention what bigbang testing is?

* + Ans: 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.
  + Big Bang testing has the advantage that everything is finished before integration testing starts.
  + The major disadvantage is that in general it is time-consuming and difficult to trace the cause of failures because of this late integration.
  + Here all component are integrated together at once, and then tested.
  + Big Bang Integration Testing:
  + Advantages:
* Convenient for small systems.
  + Disadvantages:
* Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces links to be tested could be missed easily.
* 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.

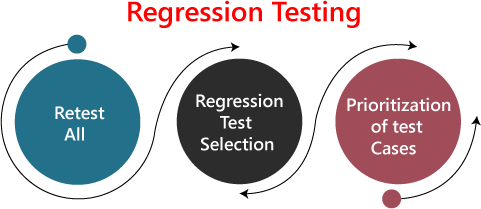
# 20. What is the purpose of exit criteria?

Ans:

* Exit criteria, also known as test completion criteria, are predefined conditions or metrics that must be met in order to determine when to stop testing activities for a particular phase, level, or project. The purpose of exit criteria is to provide a set of guidelines or benchmarks to evaluate whether testing has been performed adequately and whether the software system is ready to progress to the next phase or be released
* Here are the main purposes of exit criteria in software testing:
* **Evaluation of Test Completion:** Exit criteria help in determining when testing activities for a specific phase or level are considered complete. By defining specific conditions that need to be met, such as achieving a certain level of test coverage or executing a specified number of test cases, exit criteria enable testers and stakeholders to assess the progress and completeness of testing efforts.
* **2. Quality Assessment:** Exit criteria serve as quality gates for evaluating the overall quality of the software system. They help in measuring the level of defects, severity of issues, and other quality indicators. Meeting the exit criteria demonstrates that the software system meets the predefined quality standards and is ready for the next phase or release.
* **3. Risk Management:** Exit criteria aid in managing risks associated with software testing. They help identify and assess the risks that need to be mitigated before proceeding further. Exit criteria may include requirements such as achieving a certain level of test coverage in critical areas or resolving high-priority defects that pose significant risks to the system.
* **4. Decision Making:** Exit criteria provide a basis for making informed decisions about software release or progression to the next phase. By establishing clear and measurable criteria, stakeholders can make decisions on whether the system is stable enough, meets the required quality standards, and is ready for the next set of activities, such as user acceptance testing or deployment.
* **Resource Allocation:** Exit criteria assist in optimizing resource allocation by ensuring that testing activities are not prolonged indefinitely. Once the exit criteria are met, testing resources can be efficiently redirected to other critical tasks, maximizing the efficiency of the testing process.
* It is important to establish and document exit criteria at the beginning of a project or testing phase, with the involvement of key stakeholders. This ensures that there is a clear understanding of the goals and expectations, and provides a common reference point for evaluating the completion and readiness of the software system

# When should “Regression Testing” be performed?

* + Ans: Regression Testing: Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.
  + You also need to ensure that the modifications have not caused unintended side-effects elsewhere and that the modified system still meets its requirements – Regression Testing
  + In other words, regression testing is performed to ensure that the changes made to an application do not introduce new bugs or issues. It is an essential part of software development and helps ensure that the quality of the software remains high.
  + when testing bug-fix releases as part of the maintenance phase
  + It should be considered complete when agreed completion criteria for regression testing have been met
  + Regression test suites evolve over time and given that they are run frequently are ideal candidates for automation It should be applied at all Test Levels
  + Need of Regression Testing:
  + Change in requirements and code is modified according to the requirement new feature is added to the software
  + Defect fixing
  + Performance issue fix
  + Regression Testing can be carried out using following techniques:



# What is 7 key principles? Explain in detail?

* + Ans:
  + 1.Testing shows presence of Defects
  + 2. Exhaustive Testing is Impossible!
  + 3. Early Testing
  + 4. Defect Clustering
  + 5. The Pesticide Paradox
  + 6. Testing is Context Dependent
  + 7. Absence of Errors Fallacy

## Testing shows presence of Defects:

* + Testing can show that defects are present, but cannot prove that there are no defects.
  + Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
  + As we find more defects, the probability of undiscovered defects remaining in a system reduces.
  + However, Testing cannot prove that there are no defects present
  + Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not remove all defects.

## Exhaustive Testing is Impossible:

* + Testing everything including all combinations of inputs and preconditions is not possible.
  + Exhaustive testing is impossible means the software can never test at every test case.
  + If the software will test every test case, then it will take more cost, effort, etc.,
  + So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts.
  + For example: In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30 517 578 125 (515) test.
* So, accessing and managing risk is one of the most important activities and reason for testing in any project.
  + That is, we must Priorities our testing effort using a Risk Based Approach.

### Why do not Testing Everything?

* + Exhaustive testing of complex software applications:
* requires enormous resources
* is too expensive
* takes too long
* It is therefore impractical
* Need an alternative that is pragmatic, affordable, timely and provides results

## Early Testing:

* + Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.
  + Testing activities should start as early as possible in the development life cycle
  + The defect detected in the early phases of SDLC will be very less expensive.
  + Remember from our Definition of Testing, that Testing doesn’t start once the code has been written!

## Defect Clustering:

* + In a project, a small number of modules can contain most of the defects.
  + A small number of modules contain most of the defects discovered during pre- release testing, or are responsible for the most operational failures.
  + They are ‘clustered’
  + In other words, most defects found during testing are usually confined to a small number of modules
  + An important consideration in test prioritization!

## The Pesticide Paradox:

* + If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
  + To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
  + To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
  + Therefore, we must learn, create and use new tests based on new techniques to catch new bugs

## Testing is Context Dependent:

* + The testing approach depends on the context of the software developed.
  + Different kinds of sites are tested differently.
  + For example:
  + Safety – critical software is tested differently from an e-commerce site.
  + Whilst, Testing can be 50% of development costs, in NASA's Apollo program it was 80% testing
  + 3 to 10 failures per thousand lines of code (KLOC) typical for commercial software
  + 1 to 3 failures per KLOC typical for industrial software
  + 0.01 failures per KLOC for NASA Shuttle code!

## 7. Absence of Errors Fallacy:

* + If we build a system and, in doing so, find and fix defects ....
  + It doesn’t make it a good system
  + Even after defects have been resolved it may still be unusable and/or does not fulfil the users’ needs and expectations
* It is not only necessary that software is 99% bug-free but it is also mandatory to fulfill all the customer requirement

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

Ans:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N** | **Quality Assurance** | **Quality Control** | **Testing** |
| 1. | Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software  and intended requirements | Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements. | Activities which ensure the identification of bugs/error/defects in the Software. |
| 2. | Focuses on processes and procedures rather than conducting actual testing on the system. | Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process. | Focuses on actual testing |
| 3. | Process oriented activities. | Product oriented activities | Product oriented activities. |

# Difference between Smoke and Sanity?

Ans:

|  |  |  |
| --- | --- | --- |
| **No** | **Smoke testing** | **Sanity Testing** |
| 1. | Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality / bugs have been fixed |
| 2. | The objective of this testing is to verify "stability" of the system in order to the " with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order proceed to proceed with more rigorous testing |
| 3. | This testing is performed by the developers or testers | Sanity testing is usually performed by testers |
| 4. | Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |

# Difference between verification and Validation.

Ans:

|  |  |  |
| --- | --- | --- |
| No | **Verification** | **Validation** |
| 1 | Validation. is a static practice of verifying documents, design, code and program. | Validation is a dynamic mechanism of validating and testing the actual product |
| 2 | It does not involve executing the code. | it always involves executing the code. |
| 3 | Target is requirements specification, application and software architecture, high level, complete design, and database design etc. | Target is actual product-a unit, a module, a bent of integrated modules, and effective final product. |
| 4 | It is human based checking of documents and files. | It is computer based execution of program. |
| 5 | Validation uses methods like inspections, reviews, walkthroughs, and Desk-checking etc. | Validation uses methods like black box (functional)  testing, gray box testing, and white box |

# 26. Explain types of Performance testing.

Ans:

* [Load testing](https://artoftesting.com/types-of-performance-testing#Load_testing)
* [Stress testing](https://artoftesting.com/types-of-performance-testing#Stress_testing)
* [Endurance testing](https://artoftesting.com/types-of-performance-testing#Endurance_testing)
* [Spike testing](https://artoftesting.com/types-of-performance-testing#Spike_testing)
* [Volume testing](https://artoftesting.com/types-of-performance-testing#Volume_testing)

1. Load Testing:

Load testing involves simulating realistic user loads on the system to measure its behaviour and performance under normal and peak usage conditions. It helps determine how the system handles concurrent user requests, assess response times, and identify performance degradation or bottlenecks. Load testing ensures that the system can handle the expected user load without significant performance issues.

2. Stress Testing:

Stress testing involves testing the system beyond its normal operational capacity to evaluate its behaviour and performance under extreme or unusual conditions. It aims to identify the system's breaking point and understand how it recovers from failure. Stress testing typically involves increasing the load, data volume, or transaction complexity to push the system's limits. It helps assess the system's stability, robustness, and resilience.

## 3. Endurance testing

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

**Example**– For an application like Income tax filing, the application is used continuously for a very long duration by different users. In this type of application, memory management is very critical.

4. Spike Testing:

Spike testing involves testing the system's performance when it experiences sudden and significant increases or spikes in user load or transaction volume. It helps assess how the system handles sudden surges and if it can scale up to meet the increased demand. Spike testing can reveal issues related to resource allocation, concurrency, and response times during high-traffic situations.

5.Volume Testing:

Volume testing involves testing the system's performance with a large amount of data to assess its behaviour and response times under such conditions. It helps identify how the system handles data storage, retrieval, and processing. Volume testing can uncover performance issues related to database performance, disk space, caching, and data handling algorithms.

# 27. What is Error, Defect, Bug and failure?

Ans:

* In software testing, the terms error, defect, bug, and failure are related to different aspects of software quality issues. Here's an explanation of each term:

1. Error:

An error, also known as a mistake or a fault, is a human action or a misconception that produces an incorrect or unintended result. Errors are made during the software development process and can occur at any stage, including requirements gathering, design, coding, or documentation. Errors can lead to defects in the software.

2. Defect:

A defect, also known as a fault or a bug, is a flaw or deviation in the software that causes it to behave in an unintended or incorrect way. Defects are the result of errors made during software development or maintenance. They can manifest as coding mistakes, logic errors, missing or inaccurate functionality, or any other deviation from the expected behaviour. Defects can exist in various forms, such as syntax errors, logic errors, usability issues, performance issues, or security vulnerabilities.

3. Bug:

The term "bug" is often used interchangeably with "defect." Historically, the term "bug" was first coined by Grace Hopper when she found an actual moth causing a malfunction in a computer system. Today, "bug" is a colloquial term used to refer to defects or flaws in software. In practice, there is no significant difference between a bug and a defect—they both represent flaws in the software's behaviour.

4. Failure:

A failure occurs when the software does not behave as expected or does not meet the specified requirements or user expectations. It is the manifestation of a defect during the execution of the software. Failures can result from one or multiple defects or a combination of system-related factors. Failures can include crashes, incorrect outputs, system hangs, performance issues, or any other deviation from the expected behaviour. Failures are typically observed by end-users or testers during the testing or operational phases of the software.

* In summary, an error is a human action or misconception, a defect or bug is a flaw or deviation in the software caused by errors, and a failure is the manifestation of a defect during software execution. Defects or bugs are the root cause of failures, and testing activities aim to identify and report defects to prevent failures from occurring in the production environment.

# 28. Difference between Priority and Severity

Ans:

|  |  |  |
| --- | --- | --- |
| No | **Defect Severity** | **Defect Priority** |
| 1 | Defect Priority has defined the order in which the developer should resolve a defect | Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| 2 | Priority of defects is decided in consultation with the  manager/client | QA engineer determines the severity level of the defect |
| 3 | Priority is driven by business value | Severity is driven by functionality |
| 4 | Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| 5 | During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |
| 6 | Priority is categorized into three types   * Low * Medium * High | everity is categorized into five types   * Critical * Major * Moderate * Minor * Cosmetic |

# 29. What is Bug Life Cycle?

Ans:

* Bug Life Cycle, also known as Defect Life Cycle, is the process that a software bug goes through from its identification to its resolution. It outlines the various stages that a bug undergoes in its lifecycle, from the moment it is discovered until it is fixed and verified. The Bug Life Cycle helps in tracking and managing the progress of bug resolution within a software development or testing process.
* The typical stages of Bug Life Cycle are as follows:

1.New: This stage represents the initial state of a bug when it is reported or identified. At this stage, the bug is logged into a bug tracking system or an issue management tool.

2. Assigned: Once the bug is reported, it is assigned to a developer or a development team responsible for fixing it. The bug is now acknowledged and taken up for investigation.

3. Open: In this stage, the developer begins to analyse the bug, understand its cause, and reproduce it if necessary. They investigate the issue and plan for its resolution.

4. In Progress: Once the developer starts working on fixing the bug, its status is changed to "In Progress." The developer makes the necessary changes to the code or resolves the underlying issue causing the bug.

5.Fixed: When the developer completes the bug fixing process, the bug is marked as "Fixed." The necessary code changes or corrections have been made, addressing the reported issue.

6.Verified: In this stage, the fixed bug is verified by the testing team or the quality assurance team. They perform testing to ensure that the bug is indeed resolved and doesn't reoccur. If the fix is successful, the bug proceeds to the next stage.

7. Closed: Once the bug is verified and confirmed as fixed, it is marked as "Closed." The bug is considered resolved and no further action is required. It remains in the system for future reference.

8. Reopened: Sometimes, after being closed, a bug may resurface due to additional issues or not being completely resolved. In such cases, it is reopened, and the bug life cycle goes back to the "Assigned" or "Open" stage for further investigation and resolution.

# 30. Explain the difference between Functional testing and Non-functional testing

**Ans:**

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements. | Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system. |
| Functional testing is executed first. | Non-functional testing should be performed after functional testing. |
| Manual testing or automation tools can be used for functional testing. | Using tools will be effective for this testing. |
| Business requirements are the inputs to functional testing. | Performance parameters like speed, scalability are inputs to non-functional testing. |
| Functional testing describes what the product dose. | Non-functional testing describes how good the product works. |
| Easy to do manual testing. | Tough to do manual testing. |
| Types of Functional testing are:- Unit Testing Smoke Testing Sanity Testing Integration Testing White box testing Black Box testing User Acceptance testing Regression Testing | Types of Non-functional testing are:- Performance Testing Load Testing Volume Testing Stress Testing Security Testing Installation Testing Penetration Testing Compatibility Testing Migration Testing |

# 31. To create HLR & TestCase of

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

Ans:

<https://github.com/krishnamadhad09/krishna_tops_testing/blob/main/phone(insta).xlsx>

2) Facebook Login Page:

Ans:

<https://github.com/krishnamadhad09/krishna_tops_testing/blob/main/feb(hlr%2Ctestcase).xlsx>

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

Ans:

|  |  |
| --- | --- |
| **STLC** | **SDLC** |
| Stlc full form is software testing life cycle. | Sdlc full form is software development life life cycle |
| This article is about Software Testing Life Cycle (STLC), which involves a series of activities to ensure software quality goals are met and consists of six major phases: Requirement Analysis, Test Planning, Test case development, Test Environment setup, Test Execution and Test Cycle closure. | SDLC is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support. There are a number of different development models. |
| STLC Phases  1. Requirement Analysis  2. Test Planning  3. Test case development  4. Test Environment setup  5. Test Execution  6. Test Cycle closure | SDLC Phases  Requirements Collection/Gathering  Analysis  Design  Implementation  Testing  Maintenance |

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

Ans**:**

|  |  |  |
| --- | --- | --- |
| **Test scenarios** | **Test cases** | **Test script** |
| A Scenario is any functionality that can be tested. It is also called Test Condition, or Test Possibility. | Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. | A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected. |
| Test Scenario is ‘What to be tested’ | Test Case is ‘How to be tested’ | One script is written to explain how to simulate each business scenario |
| Test scenario is nothing but test procedure. | Test case consists of set of input values, execution precondition, expected Results and executed post-condition developed to cover certain test Condition. | The Test Script can be manual or automated |
| The scenarios are derived from use cases. | Test cases are derived (or written) from test scenario. | The Test Procedures Specification specifies the sequence of actions for a test, i.e. one or more Test Cases. It is also known as a Test Script |

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

Ans:

* A test plan is a document that outlines the overall approach, objectives, scope, and schedule for testing a software application or system. It serves as a roadmap for the testing process and provides guidance to the testing team on how to proceed with their activities. The primary purpose of a test plan is to ensure that the software or system is thoroughly tested to meet the desired quality standards.
* Here are some key pieces of information that should be covered in a test plan:

1. Introduction: This section provides an overview of the software or system being tested, including its purpose, objectives, and any relevant background information.

2. Test Objectives: It outlines the specific goals and objectives of the testing effort. This helps to align the testing activities with the overall project goals and ensure that the testing is focused and targeted.

3. Scope: The scope defines what aspects of the software or system will be tested and what will be excluded from the testing. It helps to set boundaries and provide a clear understanding of what is to be tested.

4. Test Strategy: The test strategy outlines the overall approach and methodologies to be employed during testing. It includes details on the types of tests to be conducted, such as functional testing, performance testing, security testing, etc., and the testing techniques to be used.

5. Test Schedule: This section provides a timeline for the testing activities, including start and end dates, milestones, and key deliverables. It helps to plan and manage the testing effort effectively.

6. Test Environment:

It describes the hardware, software, and network configurations that will be used for testing. It includes details about the test environment setup, test data requirements, and any dependencies on external systems or resources.

7. Test Deliverables: It lists the documents and artifacts that will be produced during the testing process, such as test cases, test scripts, test reports, and any other relevant documentation.

8. Test Execution: This section covers the actual execution of tests, including the responsibilities of the testing team, the test schedule, and the criteria for determining when testing is complete.

9. Test Risks and Mitigation: It identifies potential risks and challenges that may affect the testing process or the quality of the software. It also outlines the measures to mitigate those risks and minimize their impact.

10. Test Sign-Off: This section specifies the criteria and process for obtaining the necessary approvals and sign-off from stakeholders once testing is completed. It ensures that the software is ready for release or further stages of development.

11. Test Resources: It outlines the resources required for testing, including personnel, tools, equipment, and training needs. It helps in identifying and allocating the necessary resources for the testing effort.

12. Change Management: This section describes how changes or defects identified during testing will be managed, including the process for reporting, tracking, and resolving issues.

* Overall, a test plan provides a comprehensive overview of the testing process and helps ensure that testing activities are well-organized, efficient, and effective in achieving the desired quality goals

# 35. What is priority?

Ans:

* Priority is defined as the order in which the defects should be resolved. The priority status is usually set by the testing team while raising the defect against the dev team mentioning the timeframe to fix the defect. The Priority status is set based on end users requirement.
* For example: If the company logo is incorrectly placed in the company's web page then the priority is high but it is of low severity.
* Priority can be marked as either of the following states:
* **Low -**This defect can be fixed after the critical ones are fixed.
* **Medium -**The defect should be resolved in the subsequent builds.
* **High -**The defect must be resolved immediately because the defect is affecting the application to a considerable extent and the relevant modules cannot be used until it's fixed.
* **Urgent -**The defect must be resolved immediately because the defect is affecting the application or the product severely and the product cannot be used until it has been fixed.

# 36. What is severity?

Ans:

* Severity refers to the degree of impact or seriousness of a defect or issue on the functionality, performance, or usability of a software application or system. It is a measure of how severe or critical the issue is and helps in prioritizing and allocating resources for its resolution.
* When a defect is identified during testing or reported by users, it is assigned a severity level to indicate the impact it has on the software. The severity level is typically assigned based on the following scale or categories:
* Types of Severity of Bug/Defect may be divided into four categories in software testing :-
* This flaw implies that the process has been completely shut off, and no further action can be taken.
* Major − This is a significant flaw that causes the system to fail. Certain elements of the system, however, are still operational.
* Medium − It results in some unfavorable behavior, but the system remains functioning.
* Low − It won't create any serious system failures.

# 37. Bug categories are…

Ans:

* The specific bug categories may vary depending on the organization or project, but here are some common categories:

1. Functional Bugs: These bugs relate to issues in the functional behaviour of the software. They occur when the software does not perform its intended functions correctly or produces incorrect results. Examples include incorrect calculations, data validation failures, or incorrect handling of user inputs.
2. UI/UX Bugs: These bugs involve issues related to the user interface (UI) or user experience (UX) of the software. They may include problems with layout, formatting, responsiveness, usability, or inconsistencies in visual design. Examples include misaligned elements, overlapping text, non-responsive buttons, or confusing navigation.
3. Performance Bugs: Performance bugs refer to issues related to the performance of the software, such as slow response times, excessive memory usage, or high CPU utilization. These bugs can impact the efficiency and responsiveness of the system and may cause delays or unresponsiveness during normal usage.
4. Compatibility Bugs: Compatibility bugs occur when the software does not function properly or has issues when used in specific environments, configurations, or with certain hardware, software, or operating systems. These bugs may cause the software to crash, freeze, or produce incorrect results in specific setups.
5. Security Bugs: Security bugs involve vulnerabilities or weaknesses in the software that can be exploited by attackers to gain unauthorized access, manipulate data, or perform malicious actions. These bugs may include issues like cross-site scripting (XSS), SQL injection, authentication bypass, or insecure data storage.
6. Localization/Internationalization Bugs: These bugs are related to issues in adapting the software for different languages, cultures, or locales. They may involve problems with text encoding, translation errors, date/time format issues, or improper handling of international characters.
7. Documentation Bugs: Documentation bugs pertain to errors or omissions in the software's documentation or user guides. These bugs may include inaccurate instructions, out-dated information, missing explanations, or inconsistencies in the documentation.
8. Installation/Deployment Bugs: These bugs occur during the installation or deployment process of the software. They may involve issues like incomplete installations, incorrect dependencies, configuration problems, or compatibility issues with specific deployment environments.
9. Data Integrity Bugs: Data integrity bugs involve issues related to the accuracy, completeness, or consistency of data handled by the software. These bugs may result in data corruption, loss, or incorrect processing of information.

# 38. Advantage of Bugzila.

Ans:

* Bugzilla is an open-source issue/bug tracking system that allows developers effectively to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.
* Bugzilla is a defect tracking tool, however it can be used as a test management tool as such it can be easily linked with other test case management tools like Quality Centre, Test link etc.
* This open bug-tracker enables users to stay connected with their clients or employees, to communicate about problems effectively throughout the data management chain
* Key features of Bugzilla includes
* Advanced search capabilities
* E-mail Notifications
* Modify/file Bugs by e-mail
* Time tracking
* Strong security
* Customization

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

Ans:

* The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating
* The different Methodologies in Agile Development Model
* Scrum
* Kanban
* DSDM (Dynamic Software Development Method)
* FDD (Feature Driven Development)

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

Ans:

* Authorization and authentication are two essential concepts in web testing that relate to user access and security. Here's an explanation of each term and the differences between them:

1. Authentication: Authentication is the process of verifying the identity of a user or system before granting access to specific resources or functionalities. It ensures that the user is who they claim to be. Common authentication mechanisms include usernames and passwords, biometric authentication (such as fingerprint or facial recognition), tokens, certificates, and single sign-on (SSO) solutions. In web testing, the focus is on verifying that the authentication process works correctly and securely, preventing unauthorized access.

2. Authorization: Authorization refers to the process of granting or denying access to specific resources or functionalities based on the authenticated user's permissions and privileges. It determines what actions a user can perform and what data they can access. Authorization is usually based on roles, permissions, and access control lists (ACLs). In web testing, the goal is to ensure that the authorization rules are correctly implemented, and users can only access the resources they are authorized to access.

* To summarize, authentication is about verifying the user's identity, while authorization is about granting or denying access based on that identity.
* Common problems faced in web testing include:

1. Broken Links: Links within a web application can become broken or dead over time, leading to a poor user experience. Web testing involves checking for broken links and ensuring that all links navigate correctly.

2. Cross-Browser Compatibility: Web applications need to be tested across different browsers and their versions to ensure consistent functionality and appearance. Incompatibilities may arise due to variations in rendering engines or browser-specific features.

3. Performance Issues: Web applications need to handle a large number of concurrent users efficiently. Testing for performance issues such as slow response times, bottlenecks, and scalability problems is crucial to ensure optimal user experience.

4. Security Vulnerabilities: Web applications are often targeted by malicious actors. Web testing involves identifying and addressing common security vulnerabilities, such as cross-site scripting (XSS), cross-site request forgery (CSRF), SQL injection, and insecure session management.

5. Usability Testing: Usability testing focuses on evaluating the user-friendliness of a web application. It involves assessing the application's intuitiveness, ease of navigation, clarity of instructions, and overall user satisfaction.

6. Data Validation: Input validation is critical to prevent security vulnerabilities and data integrity issues. Web testing involves checking for proper validation of user input to prevent malicious input or unintended errors.

7. Compatibility with Different Devices: With the wide variety of devices used to access the web, including smartphones, tablets, and desktops, web testing should include testing on various screen sizes, resolutions, and operating systems to ensure compatibility and responsiveness.

* These are just a few common problems in web testing, and the actual challenges can vary de pending on the specific web application and its requirements.

# 42 To create HLR & TestCase of WebBased (WhatsApp web , Instagram)

Ans:

**1. WhatsApp Web:**

<https://github.com/krishnamadhad09/krishna_tops_testing/blob/main/wh(hlr%2Ctestcase).xlsx>

**2. Instagram web :**

<https://github.com/krishnamadhad09/krishna_tops_testing/blob/main/web(insta).xlsx>

# 44. Write a scenario of only Whatsapp chat messages

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | verify the Chat window that contains the entire chat list. |
| 2 | verify the Chat window displayed with all contacts with DP or without DP |
| 3 | verify the user can see all delivered and received messages. |
| 4 | verify the user can see the read or send time of messages. |
| 5 | verify to click on one Chat contact, and then a new window should open with history. |
| 6 | verify the Chat window displays the name of all contacts on the chat window. |
| 7 | verify the users have options like Report, Block, Clear Chat, Export Chat, and Add Shortcut. |

# 45. Write a Scenario of Pen.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | verify the types of the pen , whether  is a ball point ,gel , ink pen, etc |
| 2 | verify to check the pen can be weitten in  different papers |
| 3 | verify to check the diameter of a pen . |
| 4 | verify to check the material of the pen ,  like outer body , whether it is plastic, rubber, steel, timber, metal ,etc. |
| 5 | verify to check the golden pen it is good for  health |
| 6 | verify to check the digital pen it is benefits  because digital pen that recorded audio, video,pictures click, etc. |
| 7 | verify to check the rubber pen it is fold easily |

# 46. Write a Scenario of Pen Stand.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | check the pen stand color as per the specification |
| 2 | check the size of pen stand |
| 3 | check the pen stand stand on all land |
| 4 | check the types of shape of the pen stand |
| 5 | check the types of material of pen stand |
| 6 | check the types of test pen stand ( durability,waterproof) |

# 47. Write a Scenario of Door.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | verify the types of door wooden door,  plastic, left, car door, glass, digital etc. |
| 2 | verify check the digital doors are needs electricity |
| 3 | verify the type of locks in the door |
| 4 | Verify the number of locks in the door  interior side or exterior side |
| 5 | Verify if the door closes automatically  are so people are easily |
| 6 | Verify if the door opens inwards or  Outwards |

# 48. Write a Scenario of ATM

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | Verify the type of ATM machine, if it has a touch screen, both keypad buttons only, or both. |
| 2 | Verify that on properly inserting a valid card different banking options appear on the screen. |
| 3 | Verify that the touch of the ATM screen is smooth and operational. |
| 4 | Verify that the user is presented with the option to choose a language for further operations |
| 5 | Check that the user is asked to enter a pin number before displaying any card/bank account detail |
| 6 | Verify that there is a limited number of attempts up to which the user is allowed to enter  the pin code. |
| 7 | Verify that the user is presented with different account type options like- saving, current, etc. |
| 8 | Verify that the user is allowed to get account details like available balance. |
| 9 | Check that the correct amount of money gets withdrawn as entered by the user for cash withdrawal. |
| 10 | Check that the user cannot withdraw more amount than the total available balance and a proper message should be displayed. |

# 49. When to used Usability Testing?

Ans:

* Usability testing is typically used during the design and development process of a product or service to evaluate its usability. It involves observing and gathering feedback from users as they interact with a prototype or a finished product to identify usability issues and make informed design decisions.
* Here are some specific scenarios when usability testing is commonly used:

1. Early design stages: Usability testing can be conducted during the initial stages of the design process to gather user feedback on concept ideas, wireframes, or low-fidelity prototypes. This helps in validating design assumptions, identifying potential usability problems, and informing iterative design improvements.

2. Prototype evaluation: Usability testing is useful when testing interactive prototypes that simulate the functionality of a product before its full development. It allows designers and developers to observe how users navigate through the prototype, interact with its features, and gather insights on user expectations and preferences.

3. Iterative design: Usability testing is often conducted throughout the iterative design process to validate design changes and improvements. By testing and collecting feedback on successive versions of a product, designers can refine and optimize the user experience based on user insights and preferences.

4. Comparative evaluation: Usability testing can be used to compare different design options or variants. By testing multiple designs with users, it helps in identifying the strengths and weaknesses of each design option, making informed decisions about which design direction to pursue.

5. New feature assessment: When introducing new features or significant updates to an existing product, usability testing can gauge how well users understand and utilize the new functionality. This ensures that the new features align with user needs and expectations and do not introduce any usability problems.

6. Post-launch evaluation: Usability testing can be conducted after the product is launched to gather user feedback, identify areas of improvement, and prioritize future updates or enhancements. This helps in continuously enhancing the user experience and maintaining a user-centric approach.

* Overall, usability testing is valuable whenever you want to evaluate the usability of a product, gather user feedback, and ensure that the design meets the needs and expectations of its intended users.

# 50. What is the procedure for GUI Testing?

Ans:

* GUI testing, also known as Graphical User Interface testing, focuses on evaluating the usability and functionality of the graphical elements of a software application. Here is a general procedure for conducting GUI testing:

1. Test Planning: Define the objectives, scope, and test strategy for GUI testing. Identify the target audience and their requirements. Determine the platforms, browsers, or devices on which the application needs to be tested.

2. Identify Test Cases: Identify the test scenarios and test cases specific to GUI testing. These test cases should cover various aspects such as layout, navigation, input fields, buttons, menus, forms, error handling, and visual elements.

3. Test Environment Setup: Set up the necessary test environment, including the hardware, software, operating systems, and other dependencies required for GUI testing.

4. Functional Testing: Conduct functional testing to ensure that the GUI elements are working as intended. Test the functionality of buttons, links, menus, forms, and other interactive elements. Verify that the application responds correctly to user actions and inputs.

5. Layout and Design Testing: Evaluate the layout and design of the graphical elements. Test for consistency in color schemes, fonts, alignment, spacing, and overall visual appeal. Verify that the GUI follows established design guidelines and standards.

6. Usability Testing: Assess the usability of the GUI by simulating real user interactions. Test the application's ease of use, intuitiveness, and user-friendliness. Identify any usability issues such as confusing navigation, unclear labels, or inefficient workflows.

7. Error Handling: Test how the application handles errors and displays error messages. Verify that error messages are clear, meaningful, and provide guidance to the user on how to resolve the issue.

8. Compatibility Testing: Ensure that the GUI is compatible with different browsers, screen resolutions, and devices. Test the application on various platforms and configurations to identify any rendering or functionality issues.

9. Accessibility Testing: Evaluate the GUI for accessibility compliance. Test for compatibility with assistive technologies, keyboard navigation, alternative text for images, proper color contrast, and other accessibility guidelines.

10. Localization Testing: If the application supports multiple languages or regions, conduct localization testing to verify that the GUI elements are correctly translated and culturally appropriate for each target audience.

11. Performance Testing: Assess the GUI's performance by testing its response time, loading speed, and overall responsiveness. Check for any delays or lags in the graphical rendering or user interactions.

12. Regression Testing: Perform regression testing to ensure that any changes or fixes made to the GUI do not introduce new issues or break existing functionality.

13. Documentation: Document the test results, including any issues, bugs, or suggestions for improvement. Create a comprehensive test report that summarizes the GUI testing process and outcomes.

* Remember that the specific steps and techniques may vary depending on the project and the tools used for GUI testing. It's essential to adapt the procedure to suit the specific needs of your application and project requirements.

# 51. Write a scenario of Microwave Owen.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | Verify that the dimensions of the oven are as per the specification provided. |
| 2 | Verify that the oven’s material is optimal for its use as an oven and as per the specification. |
| 3 | Verify that the oven heats the food at the desired temperature properly. |
| 4 | Verify the ovens functioning with maximum attainable temperature. |
| 5 | Verify that the digital display is clearly visible and functions correctly. |
| 6 | Verify that the temperature regulator works correctly. |

# 52. Write a scenario of Coffee vending Machine.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | Verify that the dimension of the coffee machine is as per the specification |
| 2 | Verify that outer body, as well as inner part’s material, is as per the specification |
| 3 | Verify the input mechanism for coffee ingredients-milk, water, coffee beans/powder, etc |
| 4 | Verify that coffee should not leak when not in operation |
| 5 | Verify the amount of coffee served in single-serving is as per specification |
| 6 | Verify that the digital display displays correct information |

# 53. Write a scenario of chair.

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | verify the type of chair timber chair,  plastic, beanbag chair, digital , saadle chair, etc |
| 2 | verify check the usability of the chair  as an office chair , noraml household chair |
| 3 | verify check if there is back support in  the chair |
| 4 | verify check if there is support for hands in the chair |
| 5 | verify check the material used in making the chair wood, plastics etc |
| 6 | verify check the plastic chair are low  prise and any persone use |

# 54. To Create Scenario (Positive & Negative)

Ans:

**1. facebook Chat on Mobile**

|  |  |  |
| --- | --- | --- |
| SN | Positive Scenario | Negative Scenario |
| 1 | Verify there are minimum two Users should be available for the chat | Verify there are minimum two Users should be available for the chat or not |
| 2 | Verify there are minimum two devices (Desktop, Laptop, Phones etc) should be available. | Verify there are minimum two devices (Desktop, Laptop, Phones etc) should be available or not |
| 3 | Verify that Chat application has been installed in two devices at least. | Verify that Chat application has been installed in two devices at least or not |
| 4 | Verify that how many number of words or characters can be send in a time. | Verify that how many number of words or characters can be send in a time or not |
| 5 | Verify that User is able to create  Account in Chat application. | Verify that User is able to create Account in Chat application or not |
| 6 | Verify that User is able to sent special characters in Chat | Verify that User is able to sent special characters in Chat or not. |

**2. Gmail**

|  |  |  |
| --- | --- | --- |
| SN | Positive Scenario | Negative Scenario |
| 1 | Verify that all the read and unread emails are displayed in the inbox | Verify that all the read and unread emails are displayed in the inbox or not |
| 2 | Verify that the recently received email or unread emails are highlighted in bold in the Inbox section. | Verify that the recently received email or unread emails are highlighted in bold in the Inbox section or not |
| 3 | Verify that the recently received email has correct sender‘s name or email id, subject of the email, its preview and date or time. | Verify that the recently received email has correct sender‘s name or email id, subject of the email, its preview and date or time or not |
| 4 | Verify that the attachments can be downloaded as a single zip file. | Verify that the attachments can be downloaded as a single zip file or not |
| 5 | Verify that the attachments can be downloaded individually. | Verify that the attachments can be downloaded individually or not |

**3. Online shopping to buy product (flipkart)**

Ans:

|  |  |  |
| --- | --- | --- |
| SN | Positive Scenario | Negative Scenario |
| 1 | Verify that on the product page, the user can select the desired attribute of the product e.g. size, color, etc. | Verify that on the product page, the user can select the desired attribute of the product e.g. size, color, etc. or not |
| 2 | Verify that the user can add to the cart one or more products. | Verify that the user can add to the cart one or more products or not |
| 3 | Verify that the user can add to the cart one or more products or not | verify that the user can click the buy now button to product successfully add to cart or not |
| 4 | verify the user can see the price correctly | verify the user can see the price correctly or not |
| 5 | verify the user can apply all type of offer applied sucessfully. | verify the user can apply all type of offer applied sucessfully or not |

# 55. Write a Scenario of Wrist Watch

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | check the wrist watch type - whether it is Analog wrist watch, Digital wrist watch or Smart wrist watch. |
| 2 | check that watch shows the correct time or not on the basis of  region. |
| 3 | Check that user is able to set the time or change the time on a wrist watch. |
| 4 | Check that user is also able to change the day on wrist watch if it is available. |
| 5 | check the watch is waterproof. |

# 56. Write a Scenario of Lift (Elevator)

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | Check whether the lift can move up and down |
| 2 | Check whether the lift moves to the corresponding floor as the button of the floor is clicked |
| 3 | Check whether the lift has buttons outside to stop the lift by pressing the direction indicator button. |
| 4 | Check whether the controls/buttons in the lift are self-explanatory. |
| 5 | Check whether the lift is making noise while moving up or down. |

# 57. Write a Scenario of whatsapp Group (generate group)

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | Check if an admin can add others as Admin. |
| 2 | Check admin can remove it from the group. |
| 3 | Check the admin can delete people and add them back to the group. |
| 4 | Check every user can share information. |
| 5 | Users can share different media on WhatsApp, like photos, videos, documents, links, and simple text. |

# 58. Write a Scenario of instagram ( video call with chat )

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | check the instagram avaliable on mobile phone |
| 2 | check the instagram open succesfully on mobile phone |
| 3 | check the Camera Icon should be available on the Instagram |
| 4 | check the user can do a video call with one or more person |
| 5 | check the user can read the all chat like receiving and sending a message. |

# 59. Write a Scenario of Whatsapp payment

Ans:

|  |  |
| --- | --- |
| SN | Test Scenario |
| 1 | check the whatsapp avaliable on user mobile phone |
| 2 | check the user can tap the someone name on whatsapp app |
| 3 | check the user can tap the attchement icon |
| 4 | check the user can show the payment icon inside attchment icon |
| 5 | check the user can clearly show the proper message for send the paymeny for someone. |