Assignment - 2

Name: Yagnam Parmar

Module 2 – Manual Testing

1. What is Exploratory Testing?

   Exploratory testing is a software testing approach that is not based on predefined test cases. The tester uses their domain knowledge and experience to design and execute test cases simultaneously. This approach is effective in finding defects that are difficult to identify through traditional testing methods.

   In basic terms, exploratory testing is a manual type of software testing where testers explore and analyze a software application without any predetermined test scripts. Rather than following a precise order of steps, I just start testing and decide what to check along the way.

Functionalities are checked in a structured manner

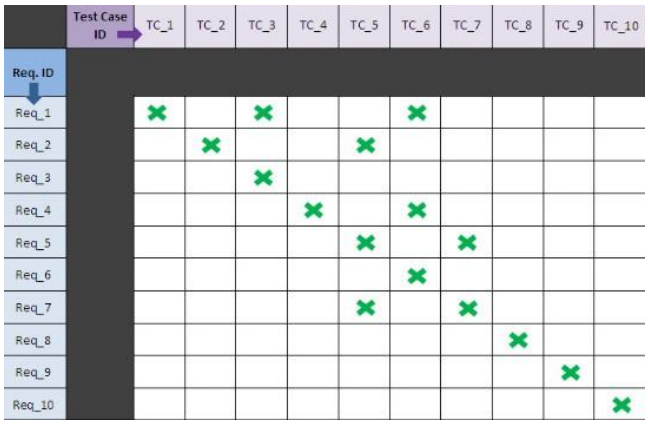
Functionalities are checked in a ad-hoc manner

Exploratory testing is a highly effective approach to testing software that gives testers the freedom to adapt and experiment on the fly based on their observations of the system and user behaviors. It allows you to unlock the full potential of your testing team and achieve better results by catching issues that rigid and traditional testing approaches might miss.

Exploratory testing lets testers use their **skills and know-how** to run tests. It helps them find problems quickly while also gathering useful feedback on the product. What sets it apart is the ability for testers to adapt their methods as they go, ensuring thorough coverage and efficient problem-solving.

2. What is traceability matrix?

A traceability matrix is a document used in project management and software development to ensure that all requirements are addressed throughout the project lifecycle. It maps requirements to their corresponding deliverables, tests, or design elements, helping teams track the relationships and ensure that all requirements are implemented and validated.



3. What is Boundary value testing?

Boundary value testing is a software testing technique used to identify errors at the boundaries rather than within the ranges of input values. The idea is that errors are more likely to occur at the edges of input domains, where values transition from one category to another.

Note:

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

Example: Consider a system that accepts ages from 18 to 56.

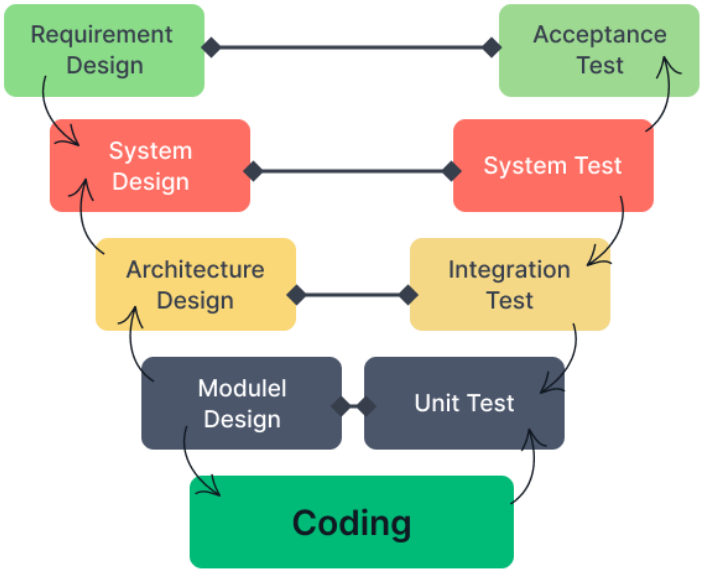
Boundary Value Analysis(Age accepts 18 to 56)		
Invalid (min-1)	Valid (min, min + 1, nominal, max – 1, max)	Invalid (max + 1)
17	18, 19, 37, 55, 56	57

4. What is Equivalence partitioning testing?

Equivalence partitioning testing is a software testing technique used to reduce the number of test cases by grouping input values into partitions or classes that are expected to produce similar results. The idea is that if one value from a partition works correctly, other values in that same partition are likely to work as well

EQUIVALENCE PARTITIONING		
Invalid	**Valid**	**Invalid**
<=17	18-56	>=57

5. What is Integration testing?



Integration testing is a Software Testing Technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

6. What determines the level of risk?

When you think of levels of risk, the low, moderate, and high terms come to mind. However, in the real software testing scenario, the risk level is determined by two dimensions: probability and impact.

Probability: It measures the likelihood of an event occurring, typically expressed as a percentage or qualitative scale. It answers the question: “How likely is it to happen?” It ranges from 0 to 100%. The probability can never be 0% because it indicates that there is no risk, and it can never be 100% as it indicates that it is no longer a risk but a certainty.

Impact: Risk, by default, brings a negative impact to any project. It assesses the consequences or severity of an event, usually quantified in monetary, temporal, or qualitative terms. It answers the question: “What would be the extent of its effect?”

RISK ASSESSMENT MATRIX

		Severity				
		1	2	3	4	5
PROBABILITY	A	Low	Low	Medium	Medium	Medium
	B	Low	Medium	Medium	High	High
	C	Medium	Medium	High	High	Very high
	D	Medium	High	High	Very high	Very high
	E	Medium	High	Very high	Very high	Very high

RISK = Severity x Probability

Case 1: Probability = D, Severity = 4
Risk = Very high

Case 2: Probability = D, Severity = 3
Risk = High

Case 3: Probability = C, Severity = 1
Risk = Medium

Case 4: Probability = A, Severity = 2
Risk = Low

By taking into account these two factors, you can calculate the level of the risk:

Level of Risk in Software = Probability of Risk Occurring X Impact of the occurred risk

Thereafter, based on the final result, the risks are classified as low, moderate, and high. Let's look at an example. Suppose a critical module of software relies on a third-party library known for occasional security vulnerabilities. Below is the assessment: development of the software.

7. What is Alpha testing?

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or the public. Alpha Testing is one of the User acceptance tests. This is referred to as Alpha testing only because it is done early on, near the end of the development of the software.

Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

1. Alpha testing is a software testing stage that takes place early in the development process, typically after the code has been written and before the final product is released to the public. Alpha testing is performed by a select group of internal stakeholders, such as developers, testers, and members of the product team.
2. The purpose of alpha testing is to identify and resolve critical bugs and issues in the software before it is released to the public. Alpha testing is performed in a controlled environment, such as a lab or a test network, and is used to simulate real-world use cases and identify any potential problems.
3. During alpha testing, the software is evaluated against a set of predetermined acceptance criteria and is tested for functionality, usability, performance, and stability. Alpha testing provides an opportunity to identify and fix bugs and issues before they reach end-users, ensuring that the final product is of high quality and meets the needs of the target audience.

8. What is beta testing?

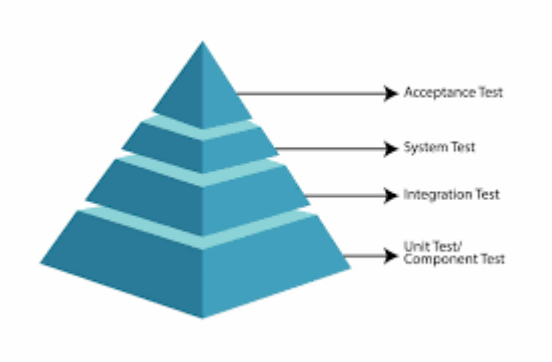
In software development, a beta test is the second phase of software testing in which a sampling of the intended audience tries out the product. Beta testing can be considered pre-release testing.

The first phase of testing in a software development lifecycle (SDLC) is known as alpha testing; *alpha* is the first letter of the Greek alphabet. This first phase includes unit testing, component testing and system testing.

Beta is the second letter of the Greek alphabet and is used to denote the second testing phase of the SDLC. Specifically, it is pre-release testing -- testing that is performed by a small group of real users in a real-world environment before the software is released to other customers or end users.

Beta testing is also sometimes referred to as user acceptance testing or end-user testing. In this phase of software development, applications are subjected to real-world testing by the intended audience for the software. The experiences of the early users are relayed to the developers, who make final changes before the software is released commercially. This feedback helps the development team improve the product, make it user-ready and minimize the risks of product failure. Alpha testing

9. What is component testing?



Component testing is a form of closed-box testing, meaning that the test evaluates the behaviour of the program without considering the details of the underlying code. Component testing is done on the section of code in its entirety, after the development has been completed.

Component testing takes longer to conduct than unit testing, because a component is made up of multiple units of code. Although it can be time-consuming, it is still very much necessary.

Sometimes the individual units work on their own but start having problems when you use them together.

Component testing examines use cases, so it could be considered a form of end-to-end(E2E) Testing. End-to-end testing and component testing replicate real-life scenarios and test the system against those scenarios from a user's perspective.

Implementing component testing is beneficial because it can reveal bugs and errors that users might encounter. It is always better to catch and fix these errors before users know about them.

10. What is functional system testing?

In functional testing, testers evaluate an application's basic functionalities against a predetermined set of specifications. Using Black box
Testing techniques, functional tests measure whether a given input returns the desired output, regardless of any other details. Results are binary: tests pass or fail.

Functional testing is a type of software testing that verifies the functionality of a software system or application by checking that ensuring that the system behaves according to the specified functional requirements and meets the intended business needs.
The goal of functional testing is to validate the system's features, capabilities, and interactions with different components. It involves testing the software's input and output, data manipulation, user interactions, and the system's response to various scenarios and conditions. Functional testing is only concerned with validating if a system works as intended.

11. What is Non-Functional Testing?

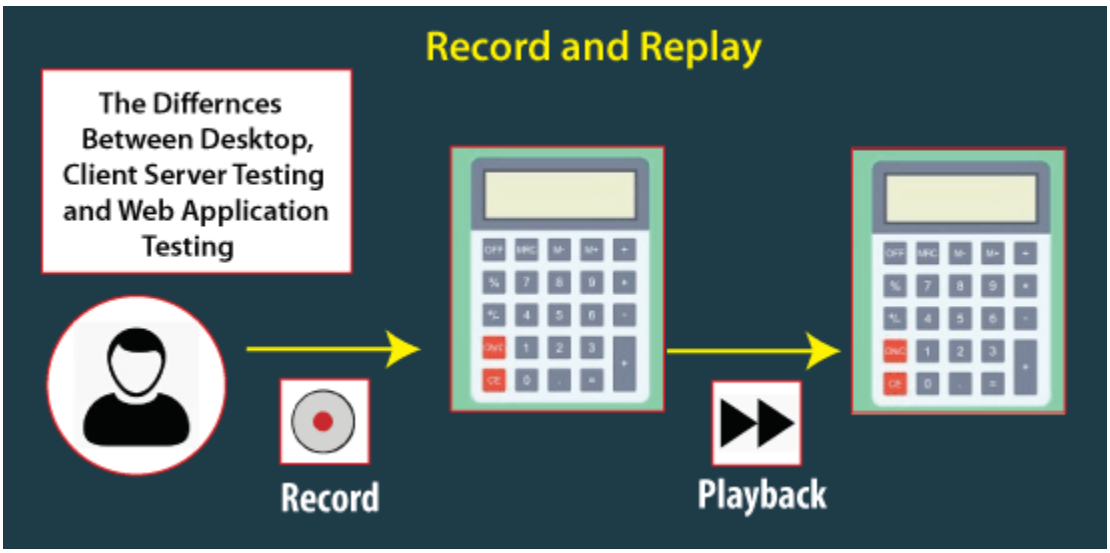
Non-functional testing is a type of testing used to evaluate a software application's performance, usability, dependability, and other non-functional characteristics.
It is intended to test a system's readiness according to non-functional criteria that functional testing never considers.
Non-Functional testing is essential for confirming the software's reliability and functionality.
The software Requirement Specification (SRS) serves as the basis for this software testing method, which enables quality assurance teams to check if the system complies with user requirements.
Increasing the product's usability, effectiveness, maintainability, and portability is the goal of non-functional testing. It aids in lowering the manufacturing risk associated with the product's non-functional components.

12. What is GUI Testing?

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

GUI is the abbreviation of 'Graphical User Interface'. It contains several visual elements, such as buttons, text boxes, menus, checkboxes, images, etc. GUI testing refers to the validating UI functions or features of an application that are visible to the users, and they should comply with business requirements. GUI testing is also known as UI Testing. That means 'User Interface testing. So, you can use both acronyms alternatively.

But why do we need GUI testing? GUI testing aims to ensure that the end-user gets a hassle-free experience. Since users are often unaware of the specific UIs, they focus on the design of the app, its colors, and whether it is easy to navigate. In this way, an app attracts more user's day by day. For this reason, GUI testing became important.



13. What is Adhoc testing?

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.

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

Adhoc testing has –

- No Documentation.
- No Test cases.
- No Test Design.

As it is not based on any test cases or requires documentation or test design resolving issues that are identified at last becomes very difficult for developers. Sometimes very interesting and unexpected errors or uncommon errors are found which would never have been found in written test cases. This Adhoc testing is used in Acceptance Tasting.

Adhoc testing saves a lot of time and one great example of Adhoc testing can be when the client needs the product by today 6 PM but the product development will be completed at 4 PM the same day. So, in hand only limited time i.e. 2 hours only, within that 2hrs the developer and tester team can test the system as a whole by taking some random inputs and can check for any errors.

Types of Adhoc Testing

1-Buddy Testing
2-Pair Testing
3-Monkey Testing

14.What is load testing?

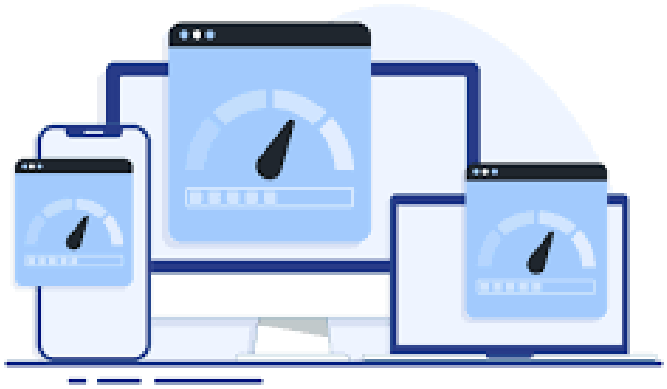
When your software development project is nearing completion, there's one test that's essential to understanding its readiness for deployment: load testing.

This type of performance Testing allows you to determine how your web application will behave during normal and peak load conditions, as well as its breaking point (should it occur below the peak load condition).

At its core, load testing is used to confirm that your web application meets your intended performance goals or objectives, which are frequently identified in a service level agreement (SLA).

More users than ever before are relying on web applications to access products or services, load testing is critical in validating that your application can function properly during realistic load scenarios. Not only does load testing mitigate the risk of your software failing, but it also mitigates the risk of your users becoming frustrated with application downtime and abandoning it altogether—which could affect your company's bottom line.

If you're unfamiliar with load testing or getting ready to perform your first one, this guide is here to help. Below, we will break down just how load testing works, as well as how you can successfully perform a load test and different load testing tools to consider.



How does load testing work?

Through specialized testing software, load testing places a simulated "load" or demand on your web application to ensure it remains stable during operation. During a load test, testing software will measure the capacity of your web application via transaction response times. If your app features extended response times or becomes unstable at a certain level of simulated traffic, your software will have likely reached its peak operating capacity—which means a solution to this software bottleneck needs to be addressed and implemented.

With load testing, development teams can easily measure and analyze things like:

- **Throughout rates, especially those required to support peak load conditions.**
- **Resource utilization levels.**
- **Hardware environment performance, such as CPU and RAM.**
- **Load balancer performance.**
- **Concurrency issues.**
- **Software functionality errors under different levels of load.**
- **Software design flaws.**
- **How many users the application can handle before breaking.**

Load testing helps developers identify issues like system lag, slow page load times, or crashes when different levels of traffic are accessing the application during production rather than post-launch. A load testing example would be a tax preparation company evaluating their web application load performance prior to peak income tax filing season and the traffic spikes it causes.

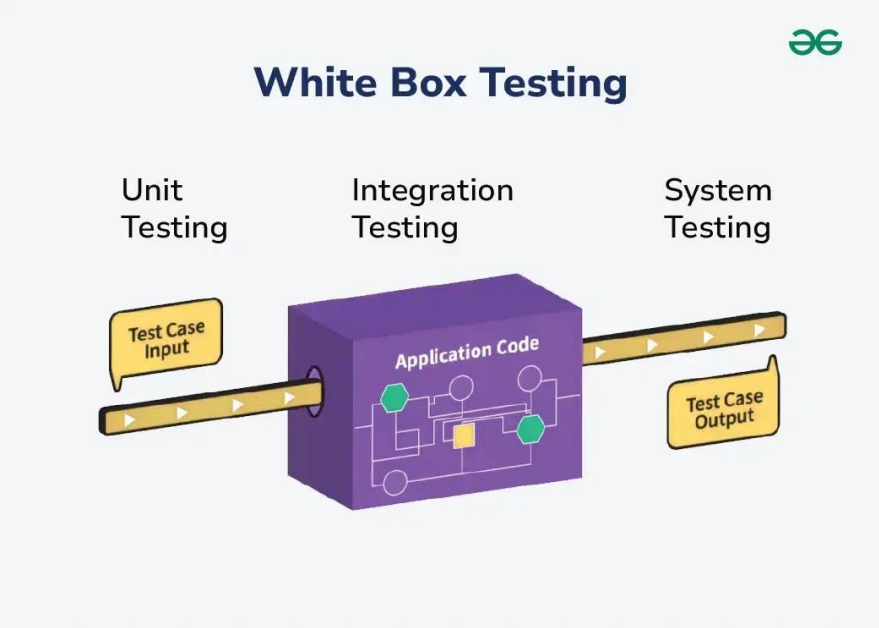
15. What is stress Testing?

Stress testing is defined as types of Software Testing that verifies the stability and reliability of the system. This test particularly determines the system's robustness and error handling under the burden of some load conditions. It tests beyond the normal operating point and analyses how the system works under extreme conditions. Stress testing is performed to ensure that the system does not crash under crunch situations. Stress testing is also known as Endurance Testing or **Torture Testing.**

- Tests used in Medicine to measure the heart's ability to respond to external stress in a controlled clinical environment .

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

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



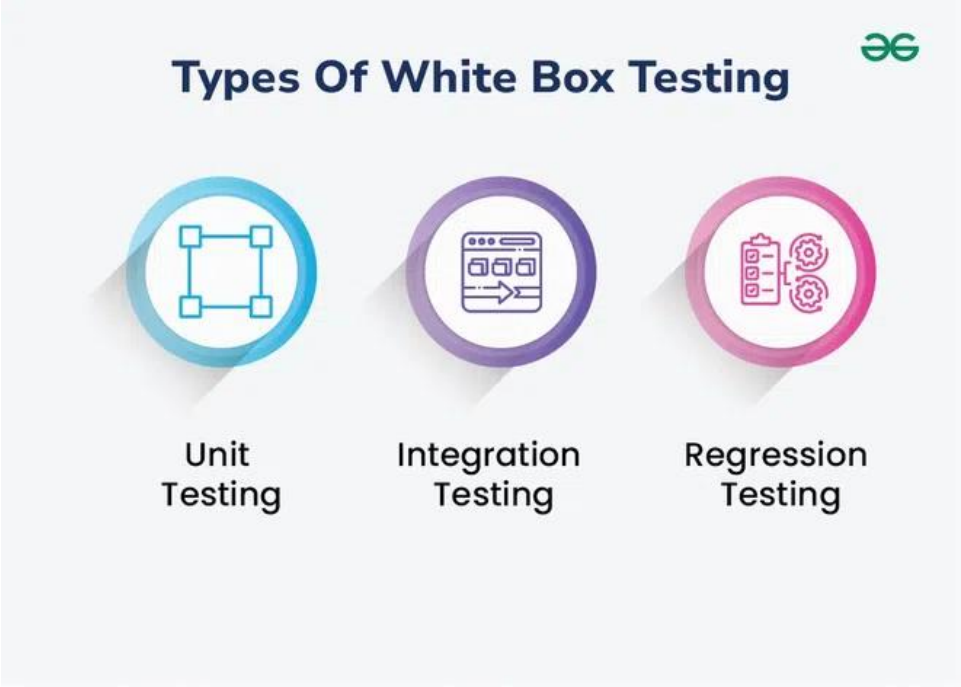
White box testing is also known as Structual Testing or code-besed Testing and it is used to test the software's internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

Before we move in depth of the white box testing do you known that their are many different type of testing used in industry and some automation testing tools are there which automate the most of testing.

Types Of White Box Testing

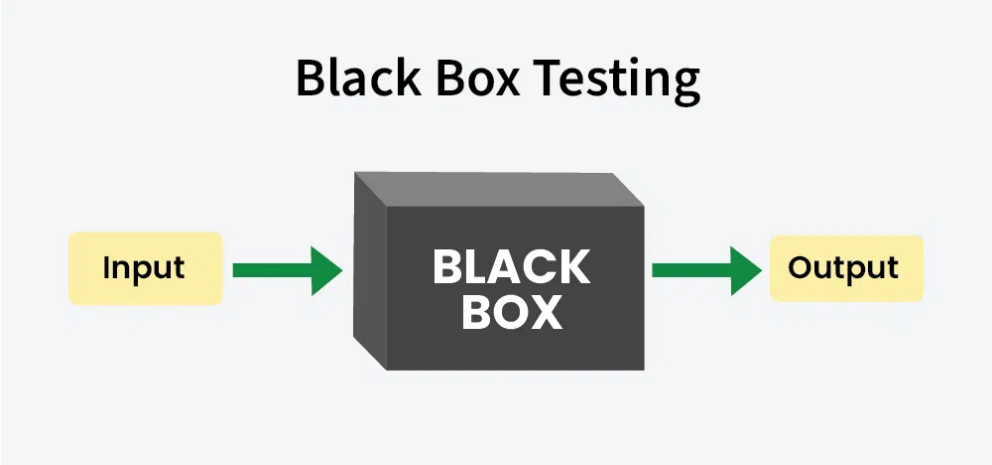
White box testing can be done for different purposes. The three main types are:

1. Unit Testing
2. Integrastion Testing
3. Regression Testing



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

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



Some black box testing techniques include:

Equivalence partitioning:
Divides input data into partitions, or equivalence classes, that are treated the same by the program. Testing one representative from each partition is usually enough to cover all potential scenarios.

Boundary value analysis:
Focuses on the values at the edges of input ranges, where errors often occur.

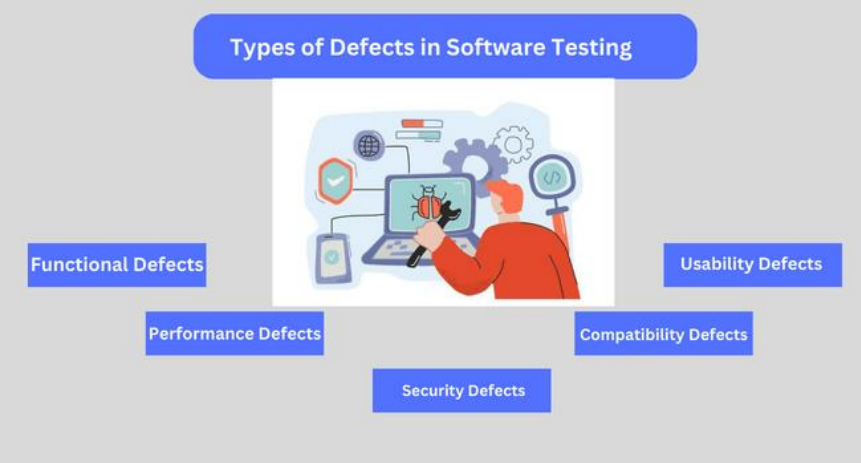
Decision table testing:
Involves creating a decision table to map different combinations of inputs to their corresponding outputs.

State transition testing:
Used when a system responds differently to the same input and depends on current conditions and previous history.

18. Mention what are the categories of defects?

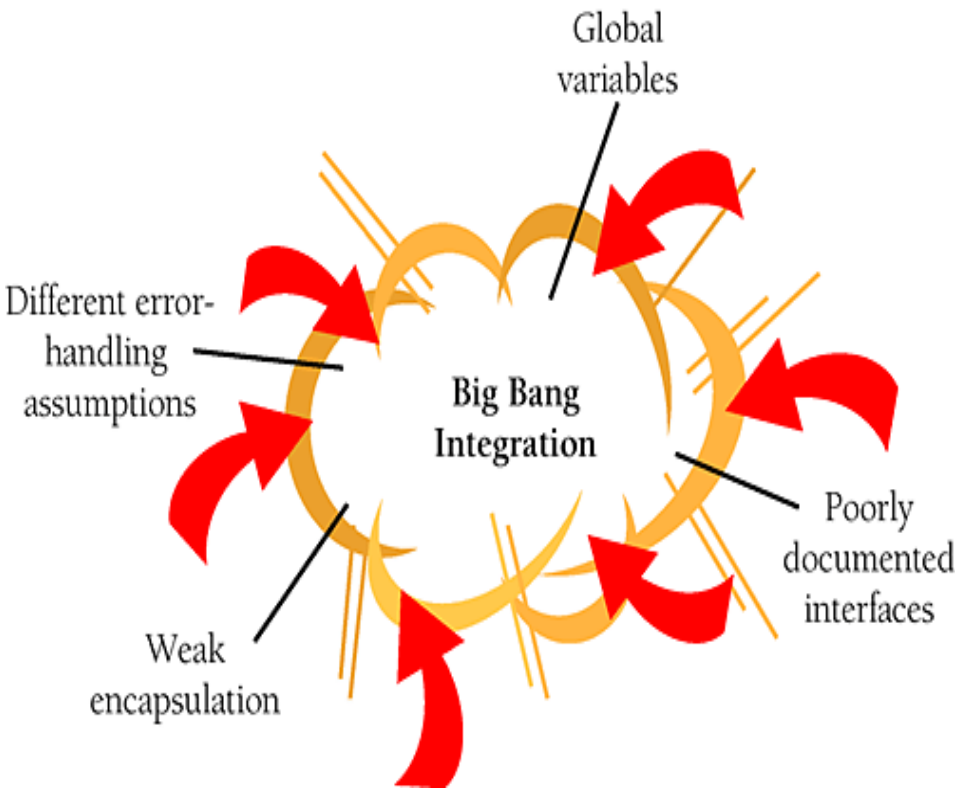
There are several ways to categorize defects, including:

- **Minor, major, and critical**
  Quality control professionals typically classify defects into these three categories based on their nature and severity. A minor defect is one that doesn't fully comply with product specs, but is still usable. A critical defect is one that non-complies with mandatory regulations and may harm the consumer's health, safety, or the environment.
- **Defect priority**
  This determines how quickly a defect needs to be fixed. Priorities are described by words, such as “critical,” “high,” “low,” or “deferred”.
- **Design defects**
  These defects occur in the design stage and can be caused by a product not being designed to meet the needs of the customer. They are also called "fatal flaws".
- **Defect severity**
  This is the impact that a defect has on either the development or execution of any program. The higher the degree of impact or severity, the more detrimental the error will be.



19. Mention what bigbang testing is?

That is a part of Integration Testing. Big Bang testing is a testing approach where all individual components or modules of a software application are tested together in a single



20. What is the purpose of exit criteria?

The main purpose of exit criteria is all condition must be full fill before leaving the project. Ensure that the project has met its objectives and requirements.

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.

21. When should "Regression Testing" be performed?

Regression testing should be performed in the following scenarios:

- When new functionality is added to the system

- When some defect is identified in the software and the code is debugged to fix it

- When the code is modified to optimize its working

- Whenever the production code is modified

- When a website's new feature is added

- When a bug is fixed by developers

- When there is an update in the database from one software to another

- After every major code change

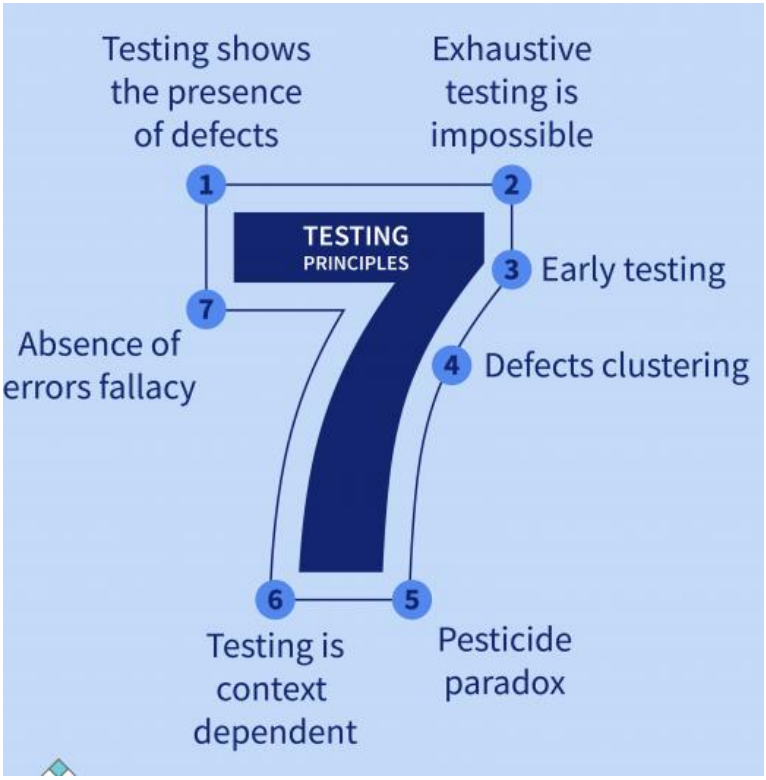
- After environment changes

- After bug fixes

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

Software testing is an important aspect of software development, ensuring that applications function correctly and meet user expectations.

In this article, we will go into the principles of software testing, exploring key concepts and methodologies to enhance product quality. From test planning to execution and analysis, understanding these principles is vital for delivering robust and reliable software solutions.



1. Testing shows the Presence of Defects

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

2. Exhaustive Testing is not Possible

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

3. Early Testing

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

4. Defect Clustering

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

5. Pesticide Paradox

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

6. Testing is Context-Dependent

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

7. Absence of Errors Fallacy

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

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

Quality Assurance	Quality Control	Tester
Focused on preventing defect and ensuring quality throughout the software development process.	Focuses on detecting and correcting defects in the software product.	Focuses on actual testing.
Quality Assurance (QA) is a process-oriented approach to ensuring that a product or service meets a set of quality standards	Quality Control (QC) is a product-oriented approach that focuses on identifying and fixing defects in the final product.	It includes activities that ensure the identification of bugs/error/defects in a software.
Involves testing, inspection, and validation to ensure the software meets the required standards.	Involves planning, designing, and implementing processes and procedures to ensure quality.	Product-oriented activities.
QA is process oriented	QC is product oriented.	Product-oriented activities.
QA is a managerial tool.	QC is a corrective tool.	

24. Difference between Smoke and Sanity?

Smoke testing	Sanity testing
To check the stability of the system.	To check the rationality of the system.
Smoke testing is scripted, documented, well-planned according to STLC	Sanity testing is unscripted, not documented, not well-planned.
This is the subset of acceptance testing	This is the subset of the regression testing.
That is performed by the developer & tester.	That is performed by tester only.
Ex. -To check the login functionality works, and the application doesn't crash or produce errors.	Ex. -To check the login functionality works as expected with a focus on the logical flow and expected behavior.

25. Difference between verification and Validation

Verification	Validation
Verification does not involve code execution	Validation involves code execution.
The verifying process includes checking documents, design, code, and program	Validation is a dynamic mechanism of testing and validating the actual product
It finds bugs early in the development cycle	It can find bug that the verification process can't find the bug
It comes before validation	It comes after verification

26. Explain types of Performance testing.

Performance testing is a type of software testing that evaluates the performance of a system or application under various conditions, such as:

- **Load:** Testing with a large number of users or transactions

- **Stress:** Testing beyond normal limits to identify breaking points

- **Endurance:** Testing for prolonged periods to evaluate stability

- **Scalability:** Testing with increased load or users to evaluate performance

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

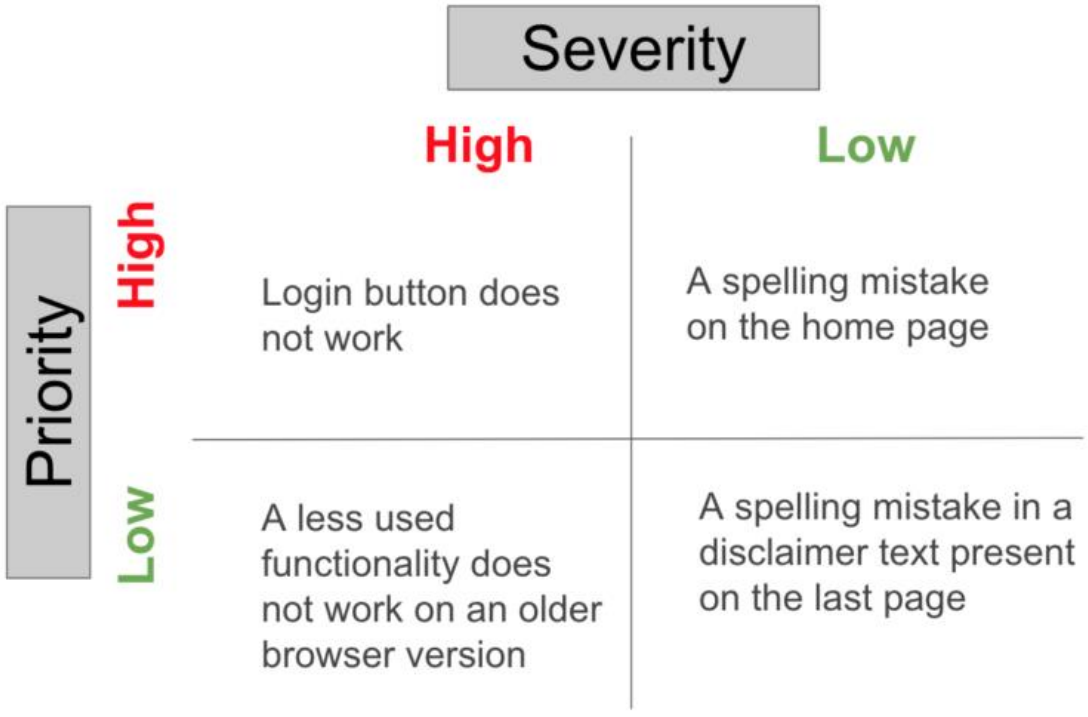
Error: A mistake is code/incorrect value

Defect: When the tester finds the error, it is called a defect.

Bug: When defect is accepted by developer or developer team, defect is called a bug.

Failure: Software failure refers to a situation where a software system or application does not perform as expected, does not meet user requirement.

28. Difference between Priority and Severity

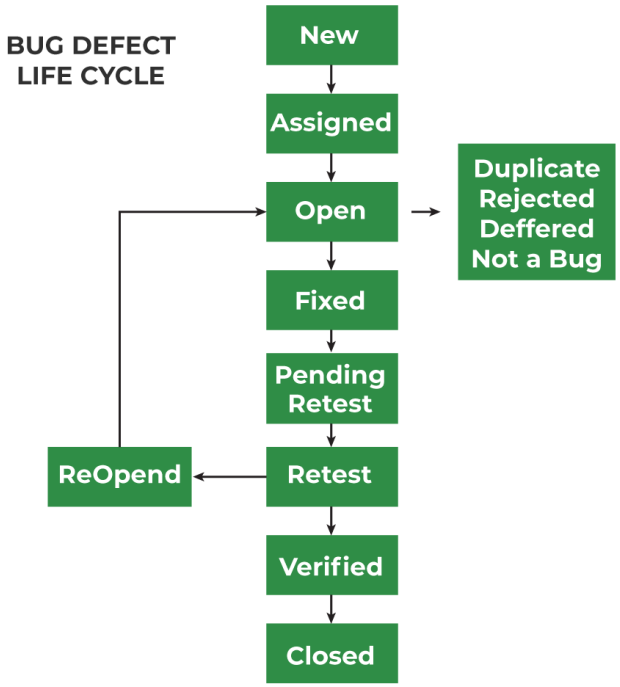


Below are the differences between severity and priority

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?

in the Software Development Process, the Defect Life Cycle is the life cycle of a defect or bug that it goes through covering a specific set of states in its entire life. Mainly bug life cycle refers to its entire state starting from a new defect detected to the closing off of that defect by the tester. Alternatively, it is also called a Bug Life Cycle.



1. New: When any new defect is identified by the Tester, it falls in the 'New' state. It is the first state of the Bug Life Cycle. The tester provides a proper Defect document to the Development team so that the development Team can refer to Defect Document and can fix the bug accordingly.

2. Assigned: Defects that are in the status of 'New' will be approved and that newly identified defect is assigned to the development team for working on the defect and to resolve that. When the defect is assigned to the developer team the status of the bug changes to the 'Assigned' state.

3. Open: In this 'Open' state the defect is being addressed by the developer team and the developer team works on the defect for fixing the bug. Based on some specific reason if the developer team feels that the defect is not appropriate then it is transferred to either the 'Rejected' or 'Deferred' state.

4. Fixed: After necessary changes of codes or after fixing identified bug developer team marks the state as 'Fixed'.

5. Pending Request: During the fixing of the defect is completed, the developer team passes the new code to the testing team for retesting. And the code/application is pending for retesting on the Tester side so the status is assigned as 'Pending Retest'.

6. Retest: At this stage, the tester starts work of retesting the defect to check whether the defect is fixed by the developer or not, and the status is marked as 'Retesting'.

7. Reopen: After 'Retesting' if the tester team found that the bug continues like previously even after the developer team has fixed the bug, then the status of the bug is again changed to 'Reopened'. Once again bug goes to the 'Open' state and goes through the life cycle again. This means it goes for Re-fixing by the developer team.

8. Verified: The tester re-tests the bug after it got fixed by the developer team and if the tester does not find any kind of defect/bug then the bug is fixed and the status assigned is 'Verified'.

9. Closed: It is the final state of the Defect Cycle, after fixing the defect by the developer team when testing found that the bug has been resolved and it does not persist then they mark the defect as a 'Closed' state.

30. Explain the difference between Functional testing and NonFunctional testing

Functional testing	**Non-Functional testing**
Ensure the software performs its intended functions correctly.	Evaluate the software's performance, security, usability, and other non-functional aspects.
Easy to do manual & automation with Functional Testing.	Tough to do manual testing with Non-functional testing.
First to execute Functional.	Non-functional should be executed after Functional testing.
Functional testing describes what the product does.	Nonfunctional testing describes how good the product works.

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

Instagram HLR & Test Case	Click Here
Facebook HLR & Test Case	Click Here

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

The STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle) are both crucial processes in software development, but they serve different purposes and focus on different aspects of the software development and testing process. Here are the key differences between STLC and SDLC:

SDLC (Software Development Life Cycle):

1. **Purpose:** SDLC is a process used by software development teams to plan, design, develop, test, and deploy software applications.
2. **Focus:** It focuses on the entire software development process, from the initial concept and requirements gathering to the final deployment and maintenance phases.
3. **Phases:** SDLC typically consists of phases such as:
    - **Requirements Gathering:** Gathering and documenting software requirements from stakeholders.
    - **Design:** Creating a detailed blueprint or design of the software architecture.
    - **Development:** Writing code and implementing the software based on the design.
    - **Testing:** Testing the software to identify and fix defects or issues.
    - **Deployment:** Releasing the software to users or customers.
    - **Maintenance:** Making updates, fixing bugs, and enhancing the software as needed after deployment.
4. **Involvement:** SDLC involves various stakeholders including developers, testers, project managers, business analysts, and end-users.
5. **Goal:** The primary goal of SDLC is to produce high-quality software that meets stakeholder requirements, within budget and time constraints.

STLC (Software Testing Life Cycle):

1. **Purpose:** STLC is a subset of the overall SDLC, specifically focusing on the testing activities throughout the software development process.
2. **Focus:** It focuses on planning, designing, executing, and managing the testing phase of the software development.
3. **Phases:** STLC typically includes phases such as:
    - **Requirement Analysis:** Analyzing the requirements from a testing perspective.
    - **Test Planning:** Creating a test plan that outlines the testing strategy, scope, resources, and schedule.
    - **Test Design:** Designing test cases and test scripts based on requirements and design documents.
    - **Test Execution:** Executing test cases, reporting defects, and retesting fixed defects.
    - **Test Closure:** Evaluating test completion criteria, preparing test closure reports, and archiving test ware.
4. **Involvement:** STLC primarily involves testers, although collaboration with developers and other stakeholders is also necessary.
5. **Goal:** The goal of STLC is to ensure that the software meets quality standards, functions correctly according to specifications, and is ready for deployment.

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

Test Scenario	Test Case	Test Script
Is any functionality that can be tested.	Is a set of action executed to verify particular feature or functionality.	Is a set of instructions to test an app automatically.
Helps to test end to end functionality in an Agile Way.	Helps in executing testing an app.	Helps to test specific things repeatedly.
Includes an end-to-end functionality to be tested.	Includes test step, data, expected result for testing.	Includes different commands to develop a script.
Allow quickly assessing the testing scope.	Allow detecting error and defects.	Allowing carrying out an automatic execution of test case.

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

A Test Plan is a detailed document outlining the approach, scope, and timeline for testing a software application or system. It serves as a guide for the testing process, ensuring that the testing is thorough, efficient, and effective.

A comprehensive Test Plan should cover the following information:

1. Test Objectives: Clearly state the goals and objectives of testing.

2. Scope: Define what is included and excluded from testing.

3. Test Environment: Describe the hardware, software, and network settings for testing.

4. Test Approach: Outline the testing methodology, techniques, and tools.

5. Test Cases: List the specific test scenarios and expected results.

6. Test Data: Identify the data required for testing and how it will be obtained.

7. Test Schedule: Provide a timeline for testing, including milestones and deadlines.

8. Resources: Identify the personnel, equipment, and budget required for testing.

9. Risks and Assumptions: Document potential risks and assumptions made during testing.

10. Test Deliverables: Define the outputs and artifacts expected from testing, such as test reports and defect logs.

11. Test Criteria: Establish the criteria for determining when testing is complete and successful.

35. To create HLR & test of Web Based (WhatsApp web, Instagram)

HLR & Test-Case of WhatsApp-web	**Click Here**
HLR & Test-Case of Instagram-web	**Click Here**
HLR & Test-Case of Art of Testing	**Click Here**

36. 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.

Types of Priorities:

Priority in software testing can be divided into 3 categories:

- **Low:** The defect is irritant but a repair can be done once the more serious defects can be fixed.
- **Medium:** The defect should be resolved during the normal course of the development but it can wait until a new version is created.
- **High:** The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

37. 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.

Types of Severity:

Severity in software testing can be classified into 4 categories:

- **Critical:** This severity level 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. However, certain parts of the system remain functional.
- **Medium:** This flaw results in unfavorable behavior but the system remains functioning.
- **Low:** This type of flaw won't cause any major breakdown in the system.

38. Bug categories are…

1. **Functional Bugs:**
2. **Performance Bugs:**
3. **Security Bugs:**
4. **Unit-level bugs:**
5. **Usability Bugs:**
6. **Syntax Errors:**
7. **Compatibility Errors:**
8. **Logic Bugs**

39. Advantage of Bugzilla

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**

40. Write a scenario of only WhatsApp chat messages

WhatsApp Chat Scenario	**Click Here**

41. Write a Scenario of Pen

Pan Scenario	Click Here

42. Write a Scenario of Pen Stand

Pan Stand Scenario	**Click Here**

43. Write a Scenario of Door

Door Scenario	Click Here

44. Write a Scenario of ATM

ATM Scenario	Click Here

45. When to used Usability Testing?

    Your work isn't done despite testing throughout the design and development cycle. You'll want to continue usability testing after a product or feature goes live to optimize consistently. This is how organizations create unique experiences that stand the test of time as user behavior evolves.

46. What is the procedure for GUI Testing?

    The procedure for GUI testing includes the following steps:

- **Identify components**: Identify the GUI components and what needs to be tested
- **Check visuals**: Verify the visual aspects of the application or website
- **Write test cases**: Write test cases to verify the working and style of the GUI components
- **Automate and test**: Automate repetitive test cases and manually test the ones that can't be automated
- **Report and retest**: Report the defects and retest again

47. Write a scenario of Microwave Owen

Microwave Owen Scenario	Click Here

48. Write a scenario of Coffee vending Machine

Coffee Vending Machin Scenario	Click Here

49. Write a scenario of chair

Chair Scenario	Click Here

50. To Create Scenario (Positive & Negative)

Gmail scenario	Click Here
Online Shopping (Flipkart)	Click Here

51. Write a Scenario of Wrist Watch

Watch Scenario	Click Here

52. Write a Scenario of Lift (Elevator)

Lift Scenario	Click here

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

WhatsApp group Scenario	Click Here

54. Write a Scenario of WhatsApp payment

WhatsApp Payment Scenario	Click Here