**What is traceability matrix?**

A traceability matrix is a document that helps to track that no requirement is missed during testing

There are 2 types of traceability matrix

1 forward traceability

2 backward traceability

**What is Boundary value testing?**

Boundary Value Testing is a software testing technique where you test the edges or boundaries of input values rather than just typical values. Since bugs often occur at boundary limits, this method helps find errors effectively.

**What is Equivalence partitioning testing?**

Equivalence Partitioning divides input data into groups (partitions) where each group is expected to behave similarly. Instead of testing all inputs, you test one value from each group to reduce test cases while ensuring good coverage.

 **Invalid (Below 18)** – e.g., 10 (should fail)

 **Valid (Between 18-60)** – e.g., 30 (should pass)

 **Invalid (Above 60)** – e.g., 70 (should fail)

**What is Integration testing?**

Integration testing is a type of software testing where different modules of an application are combined and tested to check if they work properly as a group. It ensures that data flows correctly between modules and that they function as expected when integrated.

For example, if a login page and a dashboard are developed separately, integration testing checks if, after a successful login, the user is correctly taken to the dashboard.

**What is Alpha testing?**

Alpha testing is the first phase of testing done by developers and testers before releasing the software to real users. It helps find bugs and issues in a controlled environment.

**What is beta testing?**

The user or client does beta testing after releasing software to real user

**What is component testing?**

Component testing is usually done by developers or testers. Developers test individual parts of the software during development, while testers verify if each module works correctly before integration.

**What is functional system testing?**

The function testing is a test of the actual function using of black box and wite box and end-to-end testing, and all.

**What is Non-Functional Testing?**

In the non-functional testing, there is test the process and load and stress testing and security of testing

**What is GUI Testing?**

GUI Testing checks if the visual elements of an application work correctly and look good. It ensures that buttons, icons, menus, forms, colors, and layouts function properly and match design requirements**.**

**What is Adhoc testing?**

Adhoc Testing is random testing done without a plan to find unexpected bugs. Testers explore the app freely, trying different actions to see if anything breaks or behaves wrongly.

Ther is a budy testing, Pair Testing, Monkey Testing

**What is load testing?**

in load testing, we test the software in the peak user to check if the software are working fast, stable or not.

**What is stress Testing?**

in stress testing, we try to break a software to test a software to over load user, and traffic There is compatebul for 55 users a software we test the 60 or 65 users to check the software is crashing or slowing or not

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

White box testing is a software testing technique where the internal structure, code, and logic of the application are tested. Testers check how the program processes input and produces output, ensuring everything works as expected.

Ther is types of white box testing

Statement Coverage

Decison Coverage

Condition Coverage Usabiliity Testing

**What is black box testing? What are the different black box testing techniques?**

Black box testing is a method of testing software without looking at the internal code.

**Techniques**

1. **Equivalence Partitioning** – Test by dividing inputs into groups (valid & invalid).
2. **Boundary Value Analysis (BVA)** – Check the limits (e.g., minimum and maximum values).
3. **Decision Table Testing** – Use a table to test different input and output combinations.
4. **State Transition Testing** – Test how the system moves from one state to another.
5. **Error Guessing** – Find errors based on experience and guess where issues might happen.
6. **Use Case Testing** – Test based on real-life user actions.

**Mention what are the categories of defects?**

1. **Functional Defects** – Errors in how the software functions (e.g., a login button not working).
2. **Performance Defects** – Issues related to speed, response time, or stability (e.g., slow loading pages).
3. **Usability Defects** – Problems with user experience or interface (e.g., unclear navigation, bad UI design).
4. **Compatibility Defects** – When the software does not work properly on different devices, browsers, or OS.
5. **Security Defects** – Vulnerabilities that allow unauthorized access or data breaches.
6. **Logical Defects** – Errors in calculations or incorrect business logic (e.g., wrong discount calculation).
7. **Integration Defects** – Issues when different modules or systems do not work together correctly.

**Mention what bigbang testing is?**

Big Bang Testing is a software testing approach where all the modules of an application are integrated and tested together at once, instead of testing them step by step.

**What is the purpose of exit criteria?**

1. Ensures Testing is Complete – Confirms that all planned tests are executed.
2. Defines Quality Standards – Ensures the software meets required quality levels.
3. Reduces Risk – Helps prevent releasing a buggy or incomplete product.
4. Saves Time & Resources – Avoids unnecessary testing after meeting goals.
5. Improves Decision-Making – Helps stakeholders decide when to move forward
6. We don’t have a funds to test a furthar

**When should "Regression Testing" be performed?**

Regression testing should be done whenever changes are made in the software to make sure old features still work properly.

After fixing a bug, After adding new features, After updating code, Before releasing the software

What is 7 key principles? Explain in detail?

1. **Testing Shows the Presence of Defects**

* Testing helps find **bugs** but **doesn’t** guarantee the software has **zero defects**.  
  Example: Even after testing a car, hidden issues may still exist.

1. **Exhaustive Testing is Impossible**

* It’s **impossible** to test **everything** (all inputs & scenarios).  
   Instead, testers focus on **important** areas.  
  Example: A calculator has unlimited possible inputs, so testers check only key calculations.

1. **Early Testing Saves Time & Money**

* Testing should start early to find defects sooner and fix them easily.  
  Example: Fixing a foundation issue during house construction is easier than after building the house.

1. **Defect Clustering**

* **Most bugs** are found in **a few parts** of the software.  
   Testers **focus more** on these risky areas.  
   Example: In a car, most issues may come from the **engine** rather than the mirrors or seats.

1. **Pesticide Paradox**

Repeating the same tests again and again won’t find new defects.

Testers **change test cases** to discover more bugs.

Example: Using the same pest spray daily may not kill all insects, so a **new spray** is needed

1. **Testing is Context-Dependent**

Testing **depends** on the type of software.  
 Example: A **banking app** needs high security, while a **game app** focuses on smooth graphics.

1. **Absence of Errors is a Fallacy**

Even if a system has no bugs, it might not meet user needs.  
Example: A perfect car without brakes is still useless.

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

* **QA (Quality Assurance) → Prevents problems before they happen.** Focuses on improving the process to avoid mistakes.  
  Example: Making rules so developers write better code.
* **QC (Quality Control) → Finds and fixes problems after development.** Focuses on checking the final product to ensure quality.  
   Example: Inspecting a car before selling it.
* **Tester → Checks the software to find bugs.** Runs test cases to see if everything works properly.  
  Example: A person who tests a mobile app before launch.

**Difference between Smoke and Sanity?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | |  |  |  | | --- | | **Smoke Testing** | | |  | | --- | |  |  |  | | --- | | **Sanity Testing** | |
| Basic check if software is stable for testing. | Quick check after bug fixes or updates. |
| Ensures major functions are working | Ensures recent changes work correctly. |
| |  | | --- | |  |  |  | | --- | | Before detailed testing starts. | | After a bug fix or update. |
| Covers all major features (shallow check). | Focuses only on changed parts (deep check). |
| Checking if an app opens and buttons work. | Testing only the login function after a fix. |

**Difference between verification and Validation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Verification** | |  | | --- | |  |  |  | | --- | | **Validation** | |
| Checking if the software is built correctly as per requirements. | Checking if the software meets user needs and works as expected. |
| Ensures the process of development is correct. | Ensures the final product is correct. |
| During the development phase. | |  | | --- | |  |  |  | | --- | | After the software is completely developed. | |
| Reviews, walkthroughs, inspections | Functional, usability, and performance testing. |
| Checking if the design documents and coding standards are followed. | Testing if a login feature works correctly for users. |

**Explain types of Performance testing.**

**Load –** Normal & peak users **Stress –** Beyond limit, system breakdown **Spike –** Sudden increase in users  
 **Endurance –** Long-time performance  
**Scalability –** Growth in users/resources **Volume –** Handling big data

**1. Load Testing**

* Purpose: Checks how the software works under normal and peak loads.
* Example: Testing an e-commerce site with 1,000 users at once.

**2. Stress Testing**

* Purpose: Pushes the system beyond its limits to see how it handles failures.
* Example: Testing a banking app with millions of users logging in at once.

**3. Spike Testing**

* Purpose: Checks how the system handles sudden spikes in users or traffic.
* Example: A ticket booking site getting huge traffic on a movie release day.

**4. Endurance (Soak) Testing**

* Purpose: Tests the system for a long time to find performance issues.
* Example: Running a gaming app non-stop for 24 hours to check memory leaks.

**5. Scalability Testing**

* Purpose: Checks how well the system scales when more users or resources are added.
* Example: Increasing users from 1,000 to 10,000 and seeing if the system still performs well.

**6. Volume Testing**

* Purpose: Tests how the system handles large amounts of data.
* Example: Uploading millions of records into a database to see if it crashes.

**What is Error, Defect, Bug and failure?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Error** | **Defect** | **Bug** | |  | | --- | | **Failure** |  |  | | --- | |  | |
| A mistake made by a developer while designing or coding. | A flaw or issue found during testing before release. | A defect accepted by the development team for fixing. | |  | | --- | |  |  |  | | --- | | When the software does not function correctly for users. | |
| Human mistakes in logic, syntax, or incorrect understanding. | Errors in coding or design lead to incorrect functionality. | Verified defects reported for correction. | Unfixed defects or bugs leading to real-world issues. |
| A developer writes incorrect logic, causing unexpected behavior. | A login button not working during testing due to incorrect code. | A shopping cart not adding items properly in an e-commerce app. | A banking app crashes when transferring money due to an unresolved bug. |

**Difference between Priority and Severity**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | |  |  |  | | --- | | **Priority** | | |  | | --- | |  |  |  | | --- | | **Severity** | |
| Defines how urgent a defect fix is. | |  | | --- | |  |  |  | | --- | | Defines how serious the defect impact is. | |
| |  | | --- | |  |  |  | | --- | | Business needs and timelines. | | Technical impact on the system. |
| |  | | --- | |  |  |  | | --- | | Project Manager, Client, or Business Analyst. | | Tester or Developer. |
| Affects when the defect will be fixed. | |  | | --- | |  |  |  | | --- | | Affects how much the defect affects functionality. | |
| A minor spelling mistake in the app. | A system crash or data loss issue. |

**What is Bug Life Cycle?**

The Bug Life Cycle is the process a bug goes through from being found to being fixed. Here are the simple steps:

**New → Assigned → Open → Fixed → Retest → Verified → Closed**

**1. New (Found a Bug)**

* A tester finds a problem (bug) in the software and reports it.

### ****2.Assigned (Given to a Developer)****

* The bug is assigned to a developer to fix.

### ****3️.Open (Work Starts)****

* The developer starts working on fixing the bug.

### ****4️.Fixed (Problem Solved)****

* The developer fixes the bug and marks it as "Fixed."

### ****5️.Retest (Check the Fix)****

* The tester checks if the bug is really fixed.

### ****6️.Verified (Confirmed Fixed)****

### If everything is working fine, the tester marks the bug as "Verified."

### ****7️.Closed (Bug is Gone)****

* The bug is completely fixed and closed.

**Explain the difference between Functional testing and NonFunctional testing**

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Checks what the software should do. | Checks how well the software performs. |
| Ensures the system works as expected. | Ensures quality factors like performance, security, and usability. |
| User interactions, business logic, system behavior. | Speed, security, compatibility, and efficiency. |
| Checking if a user can log in successfully. | Checking if the website loads fast with many users. |
| Unit, Integration, System, UAT, Regression Testing. | Performance, Load, Stress, Security, Usability Testing. |
| Testers, Developers. | Specialized testers (Performance, Security Testers, etc.). |
| Selenium, TestRail, QTP. | JMeter, LoadRunner, Burp Suite. |
| Must be completed before release. | Important but may be done after functional testing. |

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

|  |  |
| --- | --- |
| **SDLC (Software Development Life Cycle)** | **STLC (Software Testing Life Cycle)** |
| Process of developing software from planning to deployment. | Process of testing software to ensure it meets quality standards. |
| Focuses on designing, coding, and delivering software. | Focuses on verifying and validating software quality. |
| 1. Planning 2. Requirement Analysis 3. Design 4. Development 5. Testing 6. Deployment 7. Maintenance | 1. Requirement Analysis 2. Test Planning 3. Test Case Development 4. Test Environment Setup 5. Test Execution 6. Test Closure |
| Building the software product. | Ensuring the software is defect-free. |
| Developers, Business Analysts, Project Managers. | Testers, QA Engineers. |
| A fully developed and functional software application. | A verified and validated software with minimal defects. |
| Developing an E-commerce website. | Testing whether users can successfully complete a purchase. |

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

|  |  |  |
| --- | --- | --- |
| **Test Scenario** | **Test Case** | **Test Script** |
| A high-level description of what to test. | A detailed set of steps to perform a test. | A piece of automated code that executes a test. |
| Ensures full coverage of testing areas. | Provides step-by-step actions for testers. | Automates the test execution process. |
| General, does not include exact steps. | Includes input, expected output, and conditions. | Includes coding instructions to automate tests. |
| Testers, Business Analysts. | Testers, QA Engineers. | Automation Testers, Developers. |
| Verify that a user can log in successfully. | 1. Enter valid username and password. 2. Click the login button. 3. Verify that the homepage loads. | A Selenium script that opens a browser, enters login details, and verifies login success. |

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

A Test Plan is a document that explains how testing will be done in a project. It includes details like what to test, how to test, who will test, and when testing will happen.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Functional id | Test case name | Pre condition | Test steps | test data | Expected Result | Actual Result | Status |

**What is priority?**

Priority means how quickly a bug should be fixed based on its importance to the business. It helps developers decide which bugs to fix first.

1️.**High Priority** – Needs to be fixed **immediately** because it affects important functions.  
 Example: The **login button not working** on a banking app.

2️. **Medium Priority** – Should be fixed **soon**, but the main functions still work.  
 Example: **Wrong product image** on an e-commerce website.

3️.**Low Priority** – Can be fixed **later**, as it has a **minor impact**.  
 Example: A **spelling mistake** in a help section.

**What is severity?**

Severity means how badly a bug affects the system from a technical perspective. It helps testers decide how serious the issue is and how much it impacts functionality.

1️.**Critical Severity** – The system is **completely broken**, and nothing works.  
 Example: The app **crashes** when opened.

2️.**High Severity** – A **major function is not working**, but the system is still usable.

Example: The **payment process fails** in an e-commerce app.

3️. **Medium Severity** – A **minor function is not working**, but it does not affect major features.  
 Example: The **"Remember Me" login option** is not working.

4️. **Low Severity** – A **small issue** that does not affect functionality.  
 Example: A **spelling mistake** in the contact page

**Bug categories are**

Bug Category: Security, Database, Functionality (Critical/General), UI

**Advantage of Bugzila**

1.Free & Open Source – No licensing cost, making it budget-friendly.

2.Easy to Use – Simple interface for tracking and managing bugs.

3.Customizable – Can be modified to suit different project needs.

4.Email Notifications – Sends updates when a bug’s status changes.

5.Bug Reporting & Tracking – Allows detailed bug reports with status updates.

6.Advanced Search – Helps find specific bugs using filters.

7.Security & Access Control – Restricts access based on user roles.

8.Integration Support – Can be linked with other tools like Jira & Git.

9.Scalability – Supports large projects with multiple users.

10.Reports & Charts – Provides visual insights into bug trends.

**What are the different Methodologies in Agile Development Model?**