1.What is Exploratory Testing ?

- **Exploatory Testing** is a type of testing where testers check the software by exploring**r** it freely, without following fixed steps. Instead of using prepared test cases, they rely on their understanding and ideas to find problems.

2. What is traceability matrix ?

- A Traceability Matrix is a document that shows the relationship between requirements and test cases. It helps ensure that all are tested properly.

3. What is Boundary value testing ?

- **Boundary Value Testing** is a testing technique where you test the **edge values** or **limits** of input data. Since errors often happen at boundaries, this method helps find those issues.

4.What is Equivalence partitioning testing ?

- **Equivalence Partitioning Testing** is a technique where you divide input data into **groups** (called partitions) that are expected to behave the same. Instead of testing every value, you test **one value from each group** to save time.

5.What is Integration testing ?

- **Integration Testing** is a type of testing where different parts of a software system are combined and tested together to see if they work properly as a whole. It ensures that the interaction between different modules is correct.

6.What is Alpha testing?

- **Alpha Testing** is a type of testing done by the **developers** or **internal testers** before releasing the software to real users. It happens in the **development environment** to find and fix bugs early.

7.What is beta testing ?

- **Beta Testing** is a type of testing where **real users** test the software in a **real-world environment** before its official release. It helps find issues that developers might have missed.

8.What is component testing ?

- **Component Testing** is testing each part of a software separately to check if it works correctly before combining it with other parts.

9.What is functional system testing

- Functional System Testing (FST) is a type of software testing that evaluates whether a system meets its specified functional requirements. It verifies that the system behaves as expected by testing it as a whole, rather than focusing on individual components.

1. Key Aspects of Functional System Testing:

- End-to-End Testing – It tests the complete system, ensuring that different components interact correctly.
- Requirement Validation – Ensures the system meets functional specifications.
- Black-Box Testing – Focuses on input/output behavior without looking at internal code structure.

2 User Scenarios – Tests real-world user interactions to validate expected functionality.

3 .Functional System Testing Process:

- Requirement Analysis – Understanding functional requirements and business use cases.
- Test Planning & Case Design – Creating test cases covering different functionalities.
- Test Execution – Running test cases and comparing expected vs. actual results.
- Defect Reporting & Fixing – Identifying and reporting bugs for resolution.
- Regression Testing – Re-testing after fixes to ensure new updates don't break existing functionality.

10. What is Non-Functional Testing?

- Non-Functional Testing evaluates a system's performance, security, usability, and other quality attributes rather than its functional correctness.

Key Types:

1. **Performance Testing** – Checks speed and responsiveness.
2. **Load Testing** – Tests system behavior under expected traffic.

3. **Stress Testing** – Pushes the system to its limits.
4. **Security Testing** – Identifies vulnerabilities.
5. **Usability Testing** – Ensures user-friendliness.
6. **Scalability Testing** – Evaluates system growth handling.
7. **Compatibility Testing** – Verifies cross-platform/device functionality.

11. What is GUI Testing?

- GUI Testing checks the **visual elements** of an application to ensure they function correctly, look good, and provide a smooth user experience.
- **Key Focus Areas:**
  Layout & Design – Proper alignment, colors, fonts, responsiveness.
  Functionality – Buttons, menus, input fields, and navigation work as expected.
  Usability – User-friendly and intuitive interaction.
  Compatibility – Works across different devices, browsers, and screen sizes.
- **Types:**

- **Manual Testing** – Human testers check UI elements.
- **Automated Testing** – Uses tools like **Selenium, Appium, TestComplete** for efficiency.

12. What is Adhoc testing?

- Adhoc Testing is an informal, unstructured software testing approach where testers explore the system without predefined test cases to find defects quickly.
- Key Features:
  Unplanned & Spontaneous – No documentation or test plan.
  Exploratory – Testers use intuition and experience.
  Helps Find Critical Bugs – Often uncovers hidden defects.
  Best When Time is Limited – Used for quick checks before release.
- Example:
  A tester randomly clicks buttons and enters invalid data to see if the app crashes.

13 . What is load testing?

- Definition:
  Load Testing checks how a system performs under expected user load to ensure stability, speed, and efficiency.

- Key Focus Areas:
  1.Measures response time & scalability.
  2. Identifies performance bottlenecks.
  3. Ensures the system handles concurrent users smoothly.
- Example:
  Simulating 1,000 users on a website to check if pages load within 2 seconds.

14. What is stress Testing?

- Stress Testing is a type of performance testing that evaluates how a system behaves under extreme conditions, beyond its normal operational capacity. It checks system stability, reliability, and failure recovery under high load, excessive data, or resource exhaustion.
- Key Aspects of Stress Testing:
  - Determines the system's breaking point.
  - Identifies performance bottlenecks and failure points.
  - Ensures graceful degradation instead of a complete crash.
- Example:

Testing a banking app by simulating 1 million simultaneous transactions to check if it slows down or crashes.

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

- White Box Testing

  White Box Testing is a software testing method that examines the internal structure, code, and logic of an application. Testers have access to the source code and

- Types of White Box Testing:
  1. Unit Testing – Tests individual components or functions of the code.
  2. Integration Testing – Ensures different modules work together correctly.
  3. Mutation Testing – Modifies code slightly to check if tests detect changes.
  4. Control Flow Testing – Examines the logical flow of the program.
  5. Data Flow Testing – Checks how data moves through the system.
  6. Loop Testing – Focuses on the correctness of loops (e.g., for, while).
  7. Branch Testing – Verifies every decision point (if-else) is tested.

8. Path Testing – Ensures all possible execution paths are tested.

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

- Black Box Testing is a software testing method where the internal code, structure, and implementation details of an application are not known to the tester. The focus is on verifying functionality by providing inputs and checking expected outputs based on requirements. It is commonly used in functional and system testing.

Different Black Box Testing Techniques:

- Equivalence Partitioning – Divides input data into valid and invalid partitions to reduce the number of test cases while ensuring coverage.
- Boundary Value Analysis (BVA) – Tests input values at their minimum, maximum, just inside/outside boundaries to catch edge-case errors.
- Decision Table Testing – Uses a table to map different input combinations and their expected outputs, useful for complex logic.
- State Transition Testing – Tests how the system behaves when transitioning between different states based on inputs (e.g., login/logout, workflow changes).
- Use Case Testing – Validates real-world user scenarios to ensure the application meets business requirements.
- Error Guessing – Relies on the tester's experience and intuition to identify potential problem areas in the application.

17. Mention what are the categories of defects?

- Categories of Defects in Software Testing

1. Functional Defects – Issues related to incorrect or missing functionality (e.g., login button not working).

2. Performance Defects – System slowdowns, crashes, or failures under load (e.g., slow page load time).

3. Usability Defects – Poor UI/UX design affecting user experience (e.g., confusing navigation).

4. Compatibility Defects – Issues across different devices, browsers, or OS (e.g., website not displaying correctly on mobile).

5. Security Defects – Vulnerabilities that expose the system to threats (e.g., weak password validation).

6. Data Defects – Incorrect or inconsistent data handling (e.g., incorrect calculations or missing records).

7. Logical Defects – Errors in business logic or incorrect algorithm implementation (e.g., incorrect discount calculation).

8. Localization Defects – Issues in language, date formats, or cultural settings (e.g., incorrect currency symbols).

18. Mention what big bang testing is?

- Big Bang Testing is an integration testing approach where all system components or modules are integrated simultaneously and tested as a whole. This method is used when the entire system is ready but is risky because defects are harder to isolate.

Key Aspects:

- No incremental testing; all modules are combined at once.
- Suitable for small systems but not ideal for complex projects due to debugging challenges.
- High risk of delayed defect detection since everything is tested together.
- Example:
- In a banking system, instead of testing modules like login, transactions, and statements separately, all are integrated and tested at once.

19. What is the purpose of exit criteria?

- Purpose of Exit Criteria in Software Testing
- Exit Criteria define the conditions that must be met before testing can be stopped. It ensures the software meets quality standards and is ready for release.

- Key Purposes:
    - Ensures Testing Completeness – Confirms all planned tests are executed.
    - Defines Quality Standards – Ensures the software meets required performance, security, and usability benchmarks.
    - Prevents Premature Release – Avoids launching a product with critical defects.
    - Provides a Clear Stopping Point – Helps teams know when testing is officially complete.

- **Example Exit Criteria:**
  - 100% test case execution with a defined pass rate.
  - No critical or high-priority defects remain unresolved.
  - Performance meets the expected benchmarks.

20. When should "Regression Testing" be performed?

- Regression Testing should be performed whenever changes are made to the software to ensure that existing functionality remains unaffected.

- Key Situations for Regression Testing:

  - After Bug Fixes – To verify that fixing a defect hasn't caused new issues.
  - After Code Enhancements or Feature Additions – To ensure new features don't break existing ones.
  - After System Upgrades or Patches – To check for compatibility and stability.
  - During Continuous Integration & Deployment (CI/CD) – To maintain software reliability with frequent updates.
  - After Performance Improvements – To confirm optimizations haven't introduced unexpected behavior.

21. When should "Regression Testing" be performed?

- Regression Testing should be performed whenever changes are made to the software to ensure that existing functionality remains unaffected.

- Key Situations for Regression Testing:
  - After Bug Fixes – To verify that fixing a defect hasn't caused new issues.
  - After Code Enhancements or Feature Additions – To ensure new features don't break existing ones
  - After System Upgrades or Patches – To check for compatibility and stability.
  - During Continuous Integration & Deployment (CI/CD) – To maintain software reliability with frequent updates.
  - After Performance Improvements – To confirm optimizations haven't introduced unexpected behavior.

22 What is 7 key principles? Explain in detail?

- 7 Key Principles of Software Testing
- The 7 principles of software testing are fundamental guidelines that help ensure efficient and effective testing. These principles help testers focus on quality, risk management, and defect detection.

1. Testing Shows Presence of Defects, Not Their Absence

- Testing helps detect defects, but it cannot prove that a system is completely free of bugs.
- Even after extensive testing, some defects may still exist in the software.
  Example: A mobile app tested on multiple devices may work fine, but unforeseen issues may still arise in rare cases.

2. Exhaustive Testing is Impossible

- It is not feasible to test all possible inputs, conditions, and scenarios.
- Instead, testers use techniques like risk-based testing, equivalence partitioning, and boundary value analysis to optimize test coverage.

- Example: A banking application with multiple transaction types cannot be tested for every possible input combination, so key cases are prioritized.

3. Early Testing Saves Time and Costs

- Testing should start as early as possible in the Software Development Life Cycle (SDLC).
- Finding and fixing defects early is cheaper and prevents costly rework in later stages.
- Example: If a design flaw is identified during requirements analysis, fixing it is easier than modifying the code after development.

4. Defect Clustering (Pareto Principle – 80/20 Rule)

- A small number of modules often contain the majority of defects (80% of defects are found in 20% of the system).
- Prioritizing testing in high-risk areas improves efficiency.
- Example: In an e-commerce app, most defects might occur in payment processing rather than in static pages like the About Us section.

5. The Pesticide Paradox

- Running the same set of test cases repeatedly will no longer find new defects.
- To improve testing effectiveness, test cases should be regularly updated and new scenarios should be introduced.
- Example: If a login page is always tested with the same username-password combinations, unexpected issues might be missed.

6. Testing is Context-Dependent

- The approach and level of testing depend on the type of software, industry, and risk factors involved.
- Example: A banking application requires rigorous security and performance testing, whereas a simple blog website may only need basic functional and usability testing.

7. Absence of Errors is a Fallacy

- Even if a system has no defects, it doesn't guarantee that it meets user needs and business requirements.
- Example: A travel booking website may be bug-free but fail if it doesn't allow users to filter flights by date and price, making it useless for customers.

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

Aspect	Quality Assurance (QA)	Quality Control (QC)	Tester
Definition	Focuses on processes to prevent defects.	Focuses on product testing to detect defects.	Executes test cases to find bugs in software.
Focus	Process-oriented – Ensures correct development standards.	Product-oriented – Ensures the final product is defect-free.	Testing-oriented – Finds and reports bugs.
Goal	Prevent issues before they occur.	Identify and fix defects in the final product.	Ensure software works as expected.
Activities	Process improvement Documentation review Audits & training	Checking test results Identifying defects Verifying fixes	Writing & executing test cases Reporting bugs Performing retests
Who Performs It?	QA Engineers, Process Analysts	QC Engineers, Test Leads	Software Testers, QA Engineers

24. Difference between Smoke and Sanity?

- **Smoke**
  - Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine.
  - This testing is performed by the developers or testers.
  - Smoke testing is usually documented or scripted is unscripted.
  - Smoke testing is a subset of Regression testing .
  - smoke testing is like General Health Check

- **Sanity**
  - Sanity Testing is done to check the new functionality / bugs have been fixed
  - Sanity testing is usually performed by testers.
  - Sanity testing is usually not documented and unscripted.
  - Sanity testing is a subset of Acceptance testin
  - Sanity Testing is like specialized health checkup.

25. Difference between verification and Validation

- Definition: Ensures the right product is being built (product-oriented).

Verification	Validation
Ensures the product is being built **correctly** (process-oriented).	Ensures the **right product** is being built (product-oriented).
Checks documents, design, and processes.	Tests the actual software product.
To **prevent defects** early in development.	To **detect defects** before release.
✅ Reviewing requirements and design documents. ✅ Code reviews and walkthroughs.	Running functional/system tests. ✅ Performing UAT (User Acceptance Testing). ✅ Checking

✅ Checking test plans and cases.	software meets user needs
○	○

○

26. Explain types of Performance testing.

➢ Performance testing evaluates a system's speed stability and scalability under different conditions. Here are its .

Type	Purpose	Example
Load Testing	Checks system performance under expected load.	Simulating 1,000 users on an e-commerce site.
Stress Testing	Tests system behavior under extreme conditions.	Pushing a server beyond its limits until it crashes.
Spike Testing	Measures system response to sudden traffic surges.	Checking if a website can handle flash sales traffic spikes.
Endurance (Soak) Testing	Evaluates system stability over extended periods.	Running a web app for 24 hours to detect memory leaks.
Scalability Testing	Determines how well a system handles growth.	Uploading millions of records to test database performance.

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

Term	Definition	When It Occurs?	Example
Error	A mistake made by a developer during coding or design.	During development.	A developer writes = instead of == in a condition.
Defect	A flaw in the software found during testing.	During testing.	A login button doesn't work in testing.
Bug	A defect confirmed by developers and needs fixing.	After defect identification.	A broken link found by a tester is logged as a bug.

Failure	When the system does not work correctly in real use.	After release (live environment).	A banking app crashes when users transfer money.

Key Differences:

- **Error** → Developer's coding mistake.
- **Defect** → Issue found during testing.
- **Bug** → A defect accepted for fixing.
- **Failure** → A defect that occurs in real-world usage.

28. Difference between Priority and Severity

Aspect	Priority	Severity
Definition	Indicates how soon a defect should be fixed.	Indicates how much impact a defect has on the system.
Focus	Business urgency.	Technical impact.
Categories	High, Medium, Low.	Critical, Major, Minor, Trivial.
Example (High Priority, Low Severity)	A company logo is missing on the homepage.	Doesn't break functionality but must be fixed ASAP.
Example (Low Priority, High Severity)	A crash happens when using a rare feature.	Major issue but affects very few users.

Key Differences:

- Priority = Fix it now or later? (Business Impact)
- Severity = How bad is the issue? (Technical Impact)

29. What is Bug Life Cycle?

- **Bug Life Cycle (Defect Life Cycle)**
- The **Bug Life Cycle** is the process a defect goes through from identification to resolution. It ensures that bugs are tracked and fixed efficiently.
- **Stages of Bug Life Cycle:**

  New → The tester finds a bug and reports it.
  Assigned → The bug is assigned to a developer for fixing.
  Open → The developer starts working on the bug.

Fixed → The developer resolves the issue.
Retest → The tester verifies the fix.
Verified → If the fix works, the bug is marked as verified.
Closed → The bug is successfully fixed and closed.

- **Other Possible States:**
  - **Rejected** → If the bug is not valid or reproducible.
    Deferred → If the bug is postponed for a future release.
    Duplicate → If the same bug has already been reported.
- **Bug Life Cycle Flow:**
- **➞ New → Assigned → Open → Fixed → Retest → Verified → Closed**

30. Explain the difference between Functional testing and Non Functional testing.

Aspect	Functional Testing	Non-Functional Testing
Definition	Ensures the system works as expected based on requirements.	Evaluates system performance, security, and usability.
Focus	What the system does (features & functions).	How the system performs (quality attributes).
Testing Type	Black-box testing.	Performance, security, usability testing.
Example Tests	Login functionality. Payment processing. Data validation.	Page load speed. Security against hacking. System reliability under load.
Tools Used	Selenium, QTP, TestComplete.	JMeter, LoadRunner, AppScan.