**1. What is Exploratory Testing?**

Exploratory Testing is a dynamic and unscripted approach to software testing where testers actively explore the application to find defects, learn its behavior, and evaluate its quality at the same time.

**2. What is traceability matrix?**

A Traceability Matrix is a document used in software testing and development to ensure that all requirements are covered by test cases. It helps track the relationship between requirements and their corresponding tests, making sure nothing is missed.

**3. What is Boundary value testing?**

**Boundary Value Testing (BVT)** is a software testing technique that focuses on the **edges or boundaries** of input ranges—where errors are most likely to occur.

**4. What is Equivalence partitioning testing?**

Equivalence Partitioning Testing is a black-box testing technique used to reduce the number of test cases while maintaining effective coverage. It works by dividing input data into equivalence classes groups of values that are expected to behave similarly.

**5. What is Integration testing?**

**Integration Testing** is a level of software testing where individual modules or components are combined and tested as a group to verify that they work together correctly.

While **unit testing** checks individual pieces of code, **integration testing** ensures that those pieces interact properly when connected.

**6. What determines the level of risk?**

* **Complexity of the feature**
* **Frequency of use** by end users
* **Business criticality** (e.g., payment systems vs. UI themes)
* **Past defect history** in similar modules

**7. What is Alpha testing?**

**Alpha Testing** is a type of **internal acceptance testing** performed by the development team before the product is released to external users.

**8. What is beta testing?**

Beta Testing is a type of external user acceptance testing where a nearly finished product is released to a limited group of real users outside the development team to gather feedback before the final launch.

**9. What is beta testing?**

**Beta Testing** is a type of **external user acceptance testing** where a nearly finished product is released to a select group of real users outside the development team. The goal is to gather feedback in a **real-world environment** before the final release

**10. What is component testing?**

Component Testing, also known as module testing or unit-level testing, is a type of software testing where individual components or modules of a system are tested in isolation to verify that they function correctly.

**11.** **What is functional system testing?**

Functional System Testing is a type of software testing that verifies the complete system’s functionality against the specified requirements. It ensures that the software behaves as expected when all modules are integrated and working together.

**12.** **What is Non-Functional Testing?**

Non-Functional Testing is a type of software testing that evaluates aspects of a system that don’t relate to specific behaviors or functions, but rather to how well the system performs under various conditions.

**13.** **What is GUI Testing?**

GUI Testing (Graphical User Interface Testing) is a type of software testing that focuses on verifying the visual elements and user interactions of an application to ensure they function and appear as intended

**14. What is Adhoc Testing?**

Adhoc Testing is an informal and unstructured type of software testing performed without any predefined test cases or documentation. It relies heavily on the tester’s intuition, experience, and creativity to find defects.

**15. What is Load Testing?**

**Load Testing** is a type of **performance testing** that evaluates how a system behaves under expected or peak user loads. The goal is to ensure the software can handle the anticipated volume of traffic, data, or transactions without slowing down or crashing.

**16. What is stress Testing?**

Stress Testing is a type of non-functional testing that evaluates how a system behaves under extreme or unfavorable condition beyond its normal operating limits.

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

White Box Testing (also known as Clear Box, Glass Box, or Structural Testing) is a software testing technique where the tester has full visibility into the internal structure, logic, and code of the application. It focuses on verifying the flow of inputs through the code, checking conditions, loops, and paths to ensure everything works as intended.

Types of White Box Testing:

* **Unit Testing** – Tests individual functions or methods in isolation
* **Loop Testing** – Validates loop constructs (for, while, do-while) for correct execution
* **Branch Testing** – Ensures all decision branches (if/else, switch) are executed
* **Path Testing** – Checks all possible execution paths through the code

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

Black Box Testing is a software testing method where the tester evaluates the functionality of an application without knowing its internal code or structure. It focuses purely on inputs and outputs—what the system is supposed to do, not how it does it.

Common Black Box Testing Techniques:

* **Equivalence Partitioning** – Divides input data into valid and invalid groups; tests one value from each group
* **Boundary Value Analysis** – Focuses on edge values (min, max, just inside/outside boundaries)
* **Decision Table Testing** – Uses tables to represent combinations of inputs and expected outputs
* **State Transition Testing** – Tests system behavior when moving from one state to another (e.g., login → dashboard)
* **Error Guessing** – Relies on tester experience to guess likely error-prone areas
* **Use Case Testing** – Validates user interactions based on defined use cases or workflows

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

🧩 Categories of Defects:

* **Functional Defects** – When the system doesn’t behave as expected based on requirements
* **Performance Defects** – Issues related to speed, responsiveness, or scalability under load
* **Usability Defects** – Problems that affect user experience, navigation, or interface clarity
* **Compatibility Defects** – Errors occurring on specific devices, browsers, or operating systems
* **Security Defects** – Vulnerabilities that expose data or allow unauthorized access
* **Boundary/Edge Case Defects** – Failures at input limits or extreme values
* **Integration Defects** – Issues when modules or systems fail to interact correctly
* **Configuration Defects** – Errors due to incorrect environment setup or settings

**20.** **Mention what bigbang testing is?**

**Big Bang Testing** is an integration testing approach where **all modules or components are combined at once** and tested as a complete system, rather than integrating and testing them incrementally.

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

**Exit Criteria** are the conditions that must be met before a software testing phase (or the entire project) can be considered complete. They act as a **checkpoint** to ensure that the product is stable, meets quality standards, and is ready to move to the next stage.

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

**Regression Testing** should be performed whenever there is a **change in the codebase** to ensure that existing functionality still works as expected. Its purpose is to catch unintended side effects caused by new updates, bug fixes, or enhancements.

**23.** **What is 7 key principles? Explain in detail?**

The **7 Principles of Software Testing** are foundational guidelines that help ensure effective, efficient, and purposeful testing throughout the software development lifecycle.

The 7 Principles:

1. **Testing shows the presence of defects**
2. **Exhaustive testing is impossible**
3. **Early testing saves time and money**
4. **Defect clustering**
5. **Pesticide paradox**
6. **Testing is context dependent**
7. **Absence-of-errors fallacy**

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

| **Aspect** | **Quality Assurance (QA)** | **Quality Control (QC)** | **Tester** |
| --- | --- | --- | --- |
| **Focus** | Process-oriented: ensures proper methods are followed | Product-oriented: checks final output for defects | Execution-oriented: performs actual testing tasks |
| **Goal** | Prevent defects through process improvement | Detect defects before release | Identify and report bugs during testing |
| **Activities** | Audits, process definition, training, documentation | Reviews, inspections, validation, verification | Test case execution, bug reporting, regression tests |
| **Timing** | Throughout development lifecycle | After development, before release | During and after development phases |

**25.** **Difference between Smoke and Sanity?**

| **Aspect** | **Smoke Testing** | **Sanity Testing** |
| --- | --- | --- |
| **Purpose** | To verify basic functionality before deeper testing | To verify specific bug fixes or minor changes |
| **Scope** | Broad and shallow; covers major features | Narrow and deep; focuses on specific areas |
| **Timing** | Done after a new build is deployed | Done after bug fixes or small updates |
| **Goal** | Ensure system is stable enough for further testing | Ensure recent changes didn’t break existing functionality |

**26.** **Difference between verification and Validation?**

| **Aspect** | **Verification** | **Validation** |
| --- | --- | --- |
| **Definition** | Ensures the product is built correctly | Ensures the product meets user needs |
| **Focus** | Process-oriented | Product-oriented |
| **Activities** | Reviews, inspections, walkthroughs | Functional testing, system testing, UAT |
| **Question Answered** | "Are we building the product right?" | "Are we building the right product?" |

**27. Explain types of Performance testing?**

**⚡** Types of Performance Testing:

* Load Testing  
  Measures system performance under expected user load to ensure stability and responsiveness.
* Stress Testing  
  Pushes the system beyond its limits to identify breaking points and how it recovers from failure.
* Spike Testing  
  Tests how the system handles sudden, extreme increases in load (e.g., flash sales or viral traffic).
* Endurance Testing (Soak Testing)  
  Evaluates system performance over an extended period to detect memory leaks or degradation.
* Scalability Testing  
  Assesses the system’s ability to scale up (or down) in response to increased demand.

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

| Term | Definition |
| --- | --- |
| Error | A human mistake made during coding, design, or logic implementation |
| Defect | A deviation from expected behavior due to an error in the code or design |
| Bug | A recognized defect reported during testing or development |
| Failure | The system’s inability to perform as expected in the real-world environment |

**29.Difference between Priority and Severity**

| **Aspect** | **Priority** | **Severity** |
| --- | --- | --- |
| **Definition** | Indicates how soon a defect should be fixed | Indicates the impact of the defect on the system |
| **Focus** | Business urgency | Technical seriousness |
| **Set By** | Project manager or client | Testers or QA team |
| **Example** | A typo in the homepage title (high priority, low severity) | System crash on login (high severity, possibly low priority) |

**30. What is Bug Life Cycle?**

Bug Life Cycle Stages:

1. New – Bug is reported and logged for the first time
2. Assigned – Bug is assigned to a developer for investigation
3. Open – Developer starts analyzing and working on the fix
4. Fixed – Developer resolves the issue and marks it as fixed
5. Retest – Tester verifies the fix in the next testing cycle
6. Verified – Tester confirms the bug is resolved
7. Closed – Bug is officially closed after successful verification
8. Reopened – If the issue persists, the bug is reopened for further analysis
9. Rejected/Not a Bug – Bug is invalid or not reproducible; marked accordingly
10. Deferred – Bug is postponed for future release due to low priority or dependency

**31. Explain the difference between Functional testing and Non-Functional testing?**

| **Aspect** | **Functional Testing** | **Non-Functional Testing** |
| --- | --- | --- |
| **Definition** | Validates what the system **does** based on requirements | Validates how the system **behaves** under various conditions |
| **Focus** | Features, actions, and user interactions | Performance, usability, reliability, and scalability |
| **Techniques Used** | Black box testing, use case testing, boundary value analysis | Load testing, stress testing, security testing, usability testing |
| **Examples** | Login, registration, checkout, search functionality | Response time, memory usage, browser compatibility, user experience |

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

| **Aspect** | **SDLC (Software Development Life Cycle)** | **STLC (Software Testing Life Cycle)** |
| --- | --- | --- |
| **Purpose** | To define and manage the entire process of software development | To define and manage the process of software testing |
| **Focus** | Requirements gathering, design, coding, deployment, maintenance | Test planning, design, execution, defect tracking, closure |
| **Participants** | Developers, architects, analysts, project managers | Testers, QA engineers, test leads |
| **Outcome** | A working software product | A validated and verified software product |

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

| **Term** | **Definition** |
| --- | --- |
| **Test Scenario** | A high-level description of what to test; outlines a feature or functionality |
| **Test Case** | A detailed set of steps, inputs, and expected results to validate a scenario |
| **Test Script** | An automated or manual script that executes the test case on the application |

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

A Test Plan is a formal document that outlines the strategy, scope, resources, schedule, and objectives of the testing process for a software project. It acts as a blueprint for how testing will be conducted, ensuring clarity, consistency, and alignment across the team.

* Introduction
* Objectives & Scope
* Test Strategy
* Test Environment
* Test Deliverables
* Schedule & Milestones
* Roles & Responsibilities
* Entry & Exit Criteria
* Risk & Mitigation
* Tools Used
* Approval & Sign-off

**34. What is priority?**

Priority in software testing refers to the urgency with which a defect should be fixed. It is determined based on business needs, release schedules, and customer impact—not necessarily the technical severity of the issue.

Key Points:

* High Priority: Must be fixed immediately; affects critical business flow
* Medium Priority: Should be fixed soon but not urgent
* Low Priority: Can be fixed later; minimal impact on operation

**34. What is severity?**

**Severity** in software testing refers to the **impact** a defect has on the functionality or performance of the application. It reflects how serious the issue is from a **technical perspective**, regardless of how urgently it needs to be fixed.

**35.** **Bug categories are…**

🐞 Bug Categories:

* **Functional Bugs** – When the application doesn’t behave as expected based on requirements
* **Performance Bugs** – Issues related to speed, responsiveness, or resource usage
* **Usability Bugs** – Problems affecting user experience, navigation, or interface clarity
* **Security Bugs** – Vulnerabilities that expose data or allow unauthorized access
* **Compatibility Bugs** – Errors occurring on specific devices, browsers, or platforms
* **UI Bugs** – Visual inconsistencies like misaligned buttons, broken layouts, or font issues
* **Boundary Bugs** – Failures at input limits or edge cases
* **Integration Bugs** – Issues when modules or systems fail to interact correctly
* **Configuration Bugs** – Errors due to incorrect environment setup or system settings
* **Localization Bugs** – Mistakes in language, formatting, or cultural adaptation

**36.** **Advantage of Bugzila?**

Advantages of Bugzilla:

* **Open Source & Free**  
  No licensing cost, making it ideal for teams with budget constraints.
* **Robust Bug Tracking**  
  Allows detailed bug reporting, assignment, prioritization, and tracking.
* **Customizable Workflow**  
  Supports custom fields, statuses, and workflows to match project needs.
* **Email Notifications**  
  Sends automatic updates to stakeholders when bugs are modified or resolved.
* **Advanced Search & Reporting**  
  Powerful filters and queries help analyze defect trends and generate reports.
* **Role-Based Access Control**  
  Manages permissions for users, testers, developers, and managers.
* **Integration Support**  
  Can be integrated with version control systems and other project tools.
* **Audit Trail**  
  Maintains a history of changes for transparency and accountability.

**37.** **Difference between priority and severity**

| **Aspect** | **Priority** | **Severity** |
| --- | --- | --- |
| **Definition** | Urgency to fix the defect based on business needs | Impact of the defect on system functionality |
| **Focus** | Business importance and timelines | Technical seriousness and system disruption |
| **Decided By** | Project managers or stakeholders | Testers or QA team |
| **Example** | Typo in company name (high priority, low severity) | System crash on login (high severity, possibly low priority) |

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

**Agile Methodologies:**

| **Methodology** | **Key Focus** |
| --- | --- |
| **Scrum** | Time-boxed sprints, daily stand-ups, roles like Scrum Master and Product Owner |
| **Kanban** | Visual workflow management, continuous delivery, WIP (Work-In-Progress) limits |
| **Extreme Programming (XP)** | Frequent releases, pair programming, test-driven development (TDD) |
| **Lean** | Eliminating waste, optimizing efficiency, delivering value |
| **Crystal** | Tailored to team size and project criticality, emphasizes communication |
| **DSDM (Dynamic Systems Development Method)** | Business-driven, strict time and budget constraints |
| **Agile Unified Process (AUP)** | Simplified version of Rational Unified Process (RUP) with agile principles |
| **Feature-Driven Development (FDD)** | Model-driven, focuses on designing and building features |

**39.** **the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?**

| Aspect | Authentication | Authorization |
| --- | --- | --- |
| Definition | Verifies who the user is | Determines what the user is allowed to do |
| Purpose | Confirms identity using credentials (e.g., username/password) | Grants access to specific resources or actions |
| Occurs When | At login or sign-in | After authentication, during resource access |
| Example | Logging into a web app | Accessing admin dashboard or restricted files |

**40.** **When to used Usablity Testing?**

Usability Testing should be used when you want to evaluate how easily and effectively real users can interact with your application or product. It’s especially important during the design and development phases, and anytime you’re refining the user experience.

**41.** **What is the procedure for GUI Testing?**

1. Requirement Analysis
2. Test Planning
3. Test Case Design
4. Test Environment Setup
5. Test Execution
6. Defect Reporting
7. Regression Testing
8. Test Closer