- **Module 2(Manual Testing)**

1. What is Exploratory Testing?

Explorative Testing (also called Exploratory Testing) means testing a software application without predefined test cases, by exploring the system on the go. Testers learn about the system while testing and use their creativity, intuition, and experience to find defects.

In Simple.

Explorative Testing = Learn + Test + Discover (at the same time)

2. what is traceability matrix?

Traceability Matrix means a document or table that shows the relationship between requirements and test cases. It helps ensure that every requirement has been tested.

In Simple.

Traceability Matrix = Requirement Test Case Mapping

3. What is Boundary value testing?

Boundary Value Testing means testing at the edges (boundaries) of input values where most errors often occur.

In Simple.

We test just below, at, and just above the minimum and maximum input values

Why?

Because bugs often happen at the boundaries of input ranges, not in the middle

4. What is Equivalence partitioning testing?

Equivalence Partitioning Testing is a black-box testing technique where input data is divided into equal groups or partitions that are expected to behave the same way. From each group, only one test case is needed to represent the whole group.

In Simple.

Divide input into valid and invalid groups Test one value from each group

Why use it?

To reduce the number of test cases

To cover more was effort

5. What is Integration testing?

Integration Testing means testing how different modules or components of a software work together after they are combined.

In Simple.

Unit testing tests individual parts - Integration testing checks if they work correctly together.

Why is it important?

Because even if parts (modules) work fine alone, they might fail when connected due to:
Data mismatches
Interface errors
Wrong logic between modules

6. What determines the level of risk?

"Determines the level of risk" means to find out how dangerous or harmful something could be, and how likely it is to happen.

In Simple.

It helps us understand:
What can go wrong?
How bad will it be?
How likely is it to happen?

7. What is Alpha testing?

Alpha Testing is a type of software testing done by internal testers (like developers or QA team) before the product is released to real users

In Simple.

Alpha Testing = Testing by the company itself, not by the public.

8. What is beta testing?

Beta Testing is a type of software testing where the software is tested by real users in a real environment before final release.

In Simple.

Beta Testing = Real users test the app and give feedback

9. What is component testing?

Component Testing (also called Unit Testing) means testing individual parts (components or modules) of software separately to make sure each one works correctly on its own.

In Simple.

Component Testing = Testing one piece at a time

10. What is functional system testing?

Functional System Testing means testing the entire system to verify that all functional requirements (what the system is supposed to do) are working correctly.

Simple.

Test the whole system from the user's point of view.

Check if all features perform their functions properly.
Focus on what the system does, not how it does it.

11. What is Non-Functional Testing?

Non-Functional Testing means testing the aspects of the software that are not related to specific behaviors or functions, but rather how well the system performs under certain conditions.

In Simple.

Tests how the system works (performance, usability, reliability)
Focuses on quality attributes, not features.

12. What is GUI Testing?

GUI Testing means testing the Graphical User Interface of a software application to ensure that the interface works correctly and is user-friendly.

In Simple.

Tests buttons, menus, icons, windows, fonts, colors, layout - everything the user sees and interacts with.
Checks if all elements display properly and respond correctly to user actions.

13. What is Adhoc testing?

Adhoc Testing is an informal, unplanned testing type where testers try to find bugs by randomly exploring the software without any test cases or scripts.

In Simple.

Testing done without preparation or planning.
Tester uses creativity and intuition to find unexpected issues.
Usually done to catch bugs missed by formal testing

14. What is load testing?

Load Testing is a type of performance testing where the software is tested by applying a normal to high number of users or requests to see how well it performs under expected conditions.

In Simple.

Load Testing = Checking if the system can handle many users at once without slowing down or crashing.

15. What is stress Testing?

Stress Testing is a type of performance testing where the software is tested under extreme or beyond-normal conditions to see how it behaves when it's pushed beyond its limits.

In Simple.

Stress Testing = Testing the system under extreme pressure (too many users, too little memory, etc.) To check if it crashes or recovers properly.

Stress Testing Tools

➢ Neo Load

➢ App Perfect

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

White Box Testing (also called Structural Testing or Glass Box Testing) is a type of software testing where the internal code, logic, and structure of the program are tested directly.

In Simple.

Tester knows the internal working/code and writes tests based on how the code is written.

Types of White Box Testing:

1. Unit Testing
2. Statement Coverage
3. Branch Coverage (Decision Coverage)
4. Path Coverage
5. Condition Coverage
6. Loop Testing
7. Control Flow Testing
8. Data Flow Testing

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

Black Box Testing is a software testing method where the tester does NOT look at the internal code of the application. Instead, they test the software based on its inputs and expected outputs.

In Simple.

Tester doesn't see the code - only uses the software like a real user.

Focus is on what the system does, not how it does it.

Also called behavioral testing

Black Box Testing Techniques:

1. Equivalence Partitioning

Divide input into groups (valid and invalid) and test one value from each group.

2. Boundary Value Analysis (BVA)

Test inputs at the edges (boundaries) of valid ranges (e.g., min, max).

3. Decision Table Testing

Use a table to show all possible combinations of inputs and expected outputs.

4. State Transition Testing

Test how the system transitions from one state to another based on input (used in systems with workflows, like ATMs).

18.What is the purpose of exit criteria?

Exit Criteria are the conditions or requirements that must be met before testing can be stopped or considered complete

Purpose of Exit Criteria:

1. To Ensure Test Completeness

Confirms that enough testing has been done and major areas are covered.

2. To Ensure Quality

Makes sure that critical bugs are fixed and the software meets quality standards.

3.To Measure Progress

Acts as a checkpoint to track whether the testing goals have been achieved.

4.To Approve Software for Release

Helps decide whether the product is ready to move to the next phase (e.g., UAT or production).

19. When should "Regression Testing" be performed?

Regression Testing should be performed whenever any change is made to the software to ensure that the existing functionality still works correctly.

Specifically, it should be performed

When to perform regression testing.

1. After Bug Fixes:

2.After Code Changes:

3. After Adding New Features:

4. During Integration:

5. Before Releases:

6.After Configuration Changes:

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

The 7 Key Principles of Software Testing are fundamental guidelines that help testers plan, execute, and evaluate software tests effectively. Here they are, explained in detail

Explain in detail.

1. Testing shows presence of defects – Testing finds bugs, not proves absence.

2. Exhaustive testing is impossible – You can't test everything; focus on important areas.

3. Early testing – Start testing early to save time and cost.

4. Defect clustering – Most bugs are in a few modules.

5. Pesticide paradox – Repeating same tests won't find new bugs; update tests regularly.

6. Testing is context dependent – Testing methods vary by project type.

7. Absence of errors fallacy – Bug-free software can still fail if it doesn’t meet user needs.

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

QA (Quality Assurance)	**QC (Quality Control)**	**Tester**
Process-oriented	Product-oriented	Application or system
Prevent defects	Detect defects	Find bugs in the software
Proactive (process improvement)	Reactive (after development)	Hands-on testing (manual or automated)
Ensures correct process is followed	Verifies the final product meets requirements	Executes test cases and reports bugs
QA Engineers / Process Analysts	QC Team / Inspectors	Software Testers
Define coding standards, review processes	Review final build, validate output	Test login functionality, report login bug

In Simple.

QA = "Build it right" → ensures the process to make software is good.

QC = "Check it's built right" → ensures the final product meets expectations.

Tester = The person who actually tests the software and finds bugs.

22. Difference between Smoke and Sanity?

Smoke Testing	**Sanity Testing**
To check whether the basic functionalities work	To check specific functionality after changes or fixes
Shallow and wide	Deep and narrow
After a new build is received	After a bug fix or minor change
End-to-end major features	Only particular components or functions
To decide if the build is stable enough for further testing	To verify if a particular issue is resolved and nothing else is broken
Can the app open and login screen load?	Is the "Submit" button now working correctly after fixing?

Simple One-Liner:

Smoke Testing**:** "Is the build stable?"
Sanity Testing**:** "Is the bug really fixed?"

23. Difference between verification and Validation.

Verification	Validation
Ensures the product is built correctly (as per design/specs)	Ensures the product meets the user's actual needs
Process-oriented	Product-oriented
Done without executing the code (static testing)	Done by executing the code (dynamic testing)
Early stages of development	After development is complete
Developers, QA team	Testers, end users
Reviews, walkthroughs, inspections	Functional testing, system testing, UAT
Prevent defects	Detect defects
Reviewing requirement specs or code	Running app to check if features work for users

Summary:

- Verification = Did we build it right?
- Validation = Did we build the right thing?

24. Explain types of Performance testing.

performance testing type.

1. Load Testing – Test system under expected user load.

2. Stress Testing – Test system beyond limits to find breaking point.

3. Spike Testing – Test sudden rise/fall in user load.

4. Endurance (Soak) Testing – Test system over a long period for stability.

5. Volume Testing – Test with a large amount of data.

6. Scalability Testing – Test how well the system scales with added resources.

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

Term	Definition
Error	A human mistake made during coding, designing, or requirement understanding.
Defect	A flaw or imperfection in the software caused by an error.
Bug	A defect identified during testing.
Failure	The inability of the system or component to perform its required function.

In Simple.

- **Error** → mistake by a person
- **Defect** → issue in the code
- **Failure** → system doesn't work as expected

26. Difference between Priority and Severity.

Priority	Severity
Defines **how soon** the defect should be fixed	Defines **how serious** the defect is
Business or customer impact	Technical impact on the system
Project Manager, Product Owner, Business team	Testers or Developers
Urgency of fixing	**Impact** on functionality
High priority bugs are fixed first	High severity bugs may or may not be fixed immediately
Login page not loading = High priority	Crash in rare feature = High severity, but low priority

In Simple

- **Priority** = "How soon to fix?"
- **Severity** = "How bad is it?"

27. What is Bug Life Cycle?

Bug Life Cycle is the process a bug follows from being reported to being fixed and closed.

Stage	Description
1. New	Bug is found and reported for the first time.
2. Assigned	Bug is assigned to a developer to fix.
3. Open	Developer starts analyzing and working on the bug.
4. Fixed	Developer fixes the bug.
5. Retest	Tester retests the bug to check if the fix is working.
6. Verified	Tester confirms the bug is fixed.
7. Closed	Bug is fixed, verified, and officially closed.

28. Explain the difference between Functional testing and Non Functional testing?

Functional Testing

- Checks what the system does.
- Verifies that the software works according to requirements.
- Focuses on features and functions**.**
- Example: Login, Sign up, Add to Cart, etc.
- Types: Unit Testing, Integration Testing, System Testing, etc.
- Done using: Manual or automation tools.

Non-Functional Testing

- Checks how the system behaves.
- Verifies the software's performance and quality attributes**.**
- Focuses on speed, security, usability, reliability, etc.
- Example: How fast the login page loads, how secure the system is.
- Types: Performance Testing, Load Testing, Security Testing, etc.
- Often done using specialized tools.

One-liner:

Functional Testing checks what the system does,
Non-Functional Testing checks how well the system works.

29. 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 start to finish	Process of testing software to ensure quality
Building the software product	Validating and verifying the software
Requirement gathering, Design, Development, Testing, Deployment, Maintenance	Requirement analysis, Test planning, Test case design, Test execution, Defect tracking, Test closure
Deliver a working software product	Ensure the software meets quality standards
Developers, business analysts, project managers	Testers, QA engineers
At the beginning of the project	After the software or its module is ready for testing
Software application	Test reports, defect logs, quality metrics

In Simple

SDLC = Building the software
STLC = Testing the software

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

Term	Meaning
Test Scenario	A high-level description of what to test — the feature or functionality to be tested.
Test Case	A detailed set of conditions, inputs, and expected results to verify a particular scenario.
Test Script	An automated or manual step-by-step instruction to execute a test case or scenario.

In Simple.

- **Test Scenario** = What to test (broad idea)
- **Test Case** = How to test (detailed steps and expected results)
- **Test Script** = Automated/manual script to run the test case

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

What is a Test Plan?

A Test Plan is a formal document that defines the strategy, objectives, resources, schedule, and scope of testing activities for a software project. It guides the testing team on what to test, how to test, who will test, and when to test**.**

Information that should be covered in a Test Plan:

1. Test Plan ID
   Unique identifier for the test plan.
2. Introduction
   Overview of the project and purpose of testing.
3. Objectives
   Goals of the testing effort.
4. Scope
   Features and functionalities to be tested and not tested.
5. Test Strategy
   Overall approach to testing (types of testing, tools, techniques).
6. Test Criteria
   Entry and exit criteria for testing phases.
7. Test Deliverables
   Documents and reports to be produced (test cases, defect logs, test summary).

8. Test Environment
   Hardware, software, network, and tools required.
9. Resource Planning
   Roles and responsibilities of the testing team.
10. Schedule
    Timeline for testing activities and milestones.
11. Risk and Contingencies
    Potential risks and mitigation plans.
12. Approval
    Sign-off from stakeholders.