Module 2

Q 1. What is Exploratory Testing?

Ans. Exploratory Testing is an unscripted, hands-on approach to software testing where testers dynamically design, execute, and adapt tests in real-time based on their intuition, experience, and observations. Unlike scripted testing (where predefined test cases are followed), exploratory testing emphasizes learning, creativity, and flexibility to uncover unexpected defects.

Q 2. What is a istraceability matrix?

Ans. A Traceability Matrix is a table that connects different project items (like requirements, test cases, and bugs) to ensure nothing is missed. It helps track if all requirements are tested and if all tests cover the right requirements.

Q 3. What is Boundary value testing?

Ans. Boundary Value Testing is a software testing technique where test cases are designed to check the behavior of a system at the edges (boundaries) of input ranges.

Q 4. What is Equivalence partitioning testing?

Ans. Equivalence Partitioning is a software testing technique where input data is divided into groups (partitions) that are expected to behave the same way. Instead of testing every possible input, you test one representative value from each group to save time while maintaining coverage.

Q 5. What is Integration testing?

Ans. Integration Testing is a software testing phase where individual modules (or components) of an application are combined and tested together to ensure they work correctly as a group.

Q 6. What determines the level of risk?

Ans. Risk is a factor that could result in future negative consequences; usually expressed as impact and likelihood.

Type of Risk

1. Project Risk
2. Product Risk

Q 7. What is Alpha testing?

Ans. Alpha Testing is an early real-world testing phase where the software is tested internally (by developers or QA teams) before releasing it to external users. It happens in a controlled environment, often at the developer’s site.

Q 8. What is beta testing?

Ans. Beta Testing is a real-world testing phase where a near-final version of the software is released to a limited group of external users (outside the company) to gather feedback before the official launch.

Q 9. What is component testing?

Ans. Component Testing (also called Unit Testing or Module Testing) is a type of testing where individual parts (components) of a software application are tested in isolation to ensure they work correctly before integration.

Q 10. What is functional system testing?

Ans. Functional System Testing is a type of testing where the entire software system is tested as a whole to ensure it meets all specified functional requirements

Functional Testing Types:

1. Black Box Testing
2. White Box Testing
3. Experience based Testing
4. Smoke Testing
5. Sanity Testing
6. End to End Testing

Q 11. What is Non-Functional Testing?

Ans. Non-Functional Testing checks how well a system performs, rather than what it does. It focuses on aspects like speed, reliability, security, and usability—ensuring the software is efficient, stable, and user-friendly under real-world conditions.

Non Functional Testing Types:

1. Usability Testing
2. Compatibility Testing
3. GUI Testing
4. Security Testing
5. Performance Testing
6. Stress Testing
7. Load Testing

Q 12. What is GUI Testing?

Ans. GUI Testing (Graphical User Interface Testing) checks the visual elements of a software application to ensure they look, behave, and interact correctly with users. It focuses on buttons, menus, layouts, fonts, colors, and responsiveness.

Q 13. What is Adhoc testing?

Ans. Adhoc Testing is an informal, unplanned testing approach where testers randomly check the software without predefined test cases or documentation. It relies on the tester's experience, intuition, and creativity to find hidden bugs.

Q 14. What is load testing?

Ans. Load Testing is a type of performance testing that checks how a system behaves under expected or high user traffic. The goal is to identify bottlenecks (like slow responses or crashes) before real users experience them.

Q 15. What is stress Testing?

Ans. Stress Testing is a type of performance testing that pushes a system beyond its normal limits (e.g., extreme user traffic, high data volumes, or limited resources) to see how it handles overload and recovers from failure.

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

Ans. White Box Testing (also called Clear Box or Structural Testing) is a software testing method where the tester examines the internal code, structure, and logic of an application to design test cases. Unlike black-box testing (which focuses on functionality), white-box testing requires knowledge of programming and implementation details.

White Box Testing List:

1. Unit Tesing
2. Static Code Analysis
3. Statement Coverage
4. Branch Coverage
5. Path Coverage
6. Loop Testing
7. Mutation Testing
8. Integration Testing (White-Box Approach)

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

Ans. Black Box Testing is a software testing method where the tester evaluates the functionality of an application without knowing its internal code or structure. The focus is on inputs and outputs—testing whether the system behaves as expected based on requirements.

Black Box Testing List:

1. Equivalence Partitioning
2. Boundary Value Analysis (BVA)
3. Decision Table Testing
4. State Transition Testing

Q 18. Mention what are the categories of defects.

Ans. Based on Severity

1. Critical Defect
2. Major Defect
3. Minor Defect
4. Trivial Defect

Based on Priority

1. High Priority
2. Medium Priority
3. Low Priority

Q 19. Mention what big bang testing is?

Ans. Big Bang Testing is an integration testing approach where all or most modules of a system are combined at once and tested as a single unit, *without* incremental integration.

Example:

A team builds a basic calculator app and tests all buttons (add, subtract, etc.) together instead of checking each operation step-by-step.

Q 20. What is the purpose of exit criteria?

Ans. Exit Criteria are predefined conditions or standards that must be met before testing can be formally concluded. They ensure that the software is thoroughly validated and ready for release.

Q 21. When should "Regression Testing" be performed?

Ans. Regression Testing ensures that new code changes (like bug fixes, updates, or features) do not break existing functionality.

It should be performed in these scenarios:

1. After Code Changes
2. During Integration
3. Environment/Config Updates
4. Release Cycles
5. Periodic/Scheduled Runs

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

Ans. These principles guide testers in designing effective test strategies and avoiding common pitfalls.

1. Testing shows the presence of Defects

Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

1. Exhaustive Testing is impossible

Testing everything including all combinations of inputs and preconditions is not possible.

1. Early Testing

Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives.

1. Defect Clustering

A small number of modules contain most of the defects discovered during pre-release testing or are responsible for the most operational failures.

1. Pesticide Paradox

If the same tests are repeated over again, eventually the same set of test cases will no longer find any new defects.

1. Testing is context dependent

Testing is context-dependent. Testing is done differently in different contexts. Different kinds of sites are tested differently.

1. Absence of errors fallacy

If the system built is unusable and does not fulfil the users needs and expectations then finding and fixing defects does not help.

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

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **QA** | **QC** | **Tester** |
| Focus | Process-oriented | Product-oriented | testing functionality and behavior |
| Nature | Proactive and preventive | Reactive and corrective | Active testing |
| Objective | Ensures the process and standards are followed to prevent defects | Identifies and fixes defects in the final product | Identifies bugs and ensures quality of the product |
| Scope | Involves the entire software development process | Focuses on final product quality validation | Executes tests on the product |
| Activities | Process audits, reviews, risk management, standards creation | Test case execution, defect identification and reporting | Writing and executing test cases, reporting defects |
| Examples | Creating testing strategies, defining processes | Functional testing, performance testing | Verifying features, running test cases, reporting bugs |

Q 24. Difference between Smoke and Sanity?

Ans.

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| To verify whether the basic functionalities of the application are working correctly. | To verify if a specific functionality or bug fix works as expected after changes. |
| Broad testing of the core features of the application to check for critical failures. | Narrow testing to focus only on the changed or fixed features. |
| After receiving a new build, before deeper testing begins | After minor fixes or enhancements, typically before further detailed testing. |
| Involves a shallow and high-level check of the application's basic features | nvolves testing specific functionalities or areas impacted by recent changes. |
| Checking if the login page, homepage, and a few core functionalities are working after a build. | Verifying that a bug fix, such as the password reset function, works correctly after a recent patch. |

Q 25. Difference between verification and Validation

Ans.

|  |  |  |
| --- | --- | --- |
| **No.** | **Verification** | **Validation** |
| 1 | Verification is a process which is performed at development level | Validation is a process which is performed at testing level |
| 2 | Verification is a static testing | Validation is a dynamic testing |
| 3 | Bugs can be found during the process of development | Bugs can only be found after the process of development |
| 4 | It is used to prevent errors. | It is used to detect errors |
| 5 | Verification activities are Reviews and Inspections. | Validation activity is Testing. |
| 6 | Verification can be achieved by asking “Are you building a product right?” | Validation can be achieved by asking “Are you building a right product?” |

Q 26. Explain types of Performance testing.

Ans. Performance testing is a type of software testing that focuses on evaluating how a system performs under specific workloads. It ensures the software application is stable, scalable, and responsive. There are several types of performance testing, each designed to test different aspects of system performance:

1. Load Testing

Purpose: To check how the system behaves under an expected load.

Example: Testing a website with 1,000 users logging in at the same time.

Goal: Identify performance bottlenecks under normal conditions.

2. Stress Testing

Purpose: To determine the system's breaking point by pushing it beyond normal load limits.

Example: Doubling the expected user load to see if the system crashes.

Goal: Understand how the system behaves under extreme conditions and recoverability.

3. Spike Testing

Purpose: To test how the system reacts to sudden, large spikes in user load.

Example: Suddenly increasing the number of users from 500 to 5,000 in seconds.

Goal: Evaluate the system's ability to handle sharp changes in traffic.

4. Endurance Testing (Soak Testing)

Purpose: To test the system's performance over an extended period under a normal load.

Example: Running a test with 500 users for 24 hours.

Goal: Detect issues like memory leaks, resource exhaustion, or degradation over time.

5. Volume Testing (Flood Testing)

Purpose: To test the system’s ability to handle a large volume of data.

Example: Inputting a huge database file into the application.

Goal: Ensure the system performs well with high data volume.

6. Scalability Testing

Purpose: To determine how well the system scales when resources (like users, data, or hardware) are increased.

Example: Gradually increasing the number of users to see how system performance changes.

Goal: Find the system’s maximum capacity and optimal resource usage.

7. Configuration Testing

Purpose: To evaluate performance under different system configurations.

Example: Changing the number of CPUs or memory size and testing performance.

Goal: Identify the best configuration for optimal performance.

Q 27. What is Error, Defect, Bug and Failure?

Ans. A mistake in coding is called error, error found by tester is called defect, defect accepted by development team then is called bug, build does not meet the requirements then it is failure.

Q 28. Difference between Priority and Severity

Ans.

|  |  |
| --- | --- |
| **Functional Testing** | **Non-Functional Testing** |
| Verifies that the software functions as expected and meets business requirements. | Verifies the system's attributes such as performance, security, and usability. |
| Focuses on functional requirements. | Focuses on non-functional requirements. |
| Ensure that specific functions, features, and business logic are working correctly. | Overall system performance, security, scalability, etc. |
| Unit, Integration, System, Acceptance Testing | Performance Testing, Security Testing, Usability Testing, Compatibility Testing |
| Validates "what the system does" | Validates "how the system performs" |

Q 29. What is Bug Life Cycle?

Ans. The Bug Life Cycle describes the stages a defect goes through from discovery to resolution. It ensures proper tracking and closure of bugs in a systematic way.

Stages of the Bug Life Cycle

1. New

A bug is reported for the first time by a tester/end-user.

Example: "Login button fails on mobile devices."

2. Assigned

The bug is assigned to a developer for fixing.

3. Open

The developer starts analyzing/fixing the bug.

4. Fixed

The developer resolves the issue and marks it as fixed.

5. Retest

The tester verifies if the bug is truly fixed.

Verified → Moves to "Closed."

Not Fixed → Reopened.

6. Reopened

If the bug still exists, it goes back to the developer.

7. Closed

The bug is fully resolved and no longer exists.

8. Duplicate

If the bug is reported multiple times, it’s marked as a duplicate.

9. Rejected

The developer may reject the bug if:

It’s not reproducible.

It’s not a defect (expected behavior).

10. Deferred

The bug is postponed (e.g., for a future release).

Q 30. Explain the difference between Functional testing and non-functional testing

Ans.

Functional Testing

Functional testing focuses on what the system does. It verifies that the software functions according to the defined business requirements. This type of testing checks individual features like login, signup, form submissions, and other user interactions to make sure they work correctly.

* Purpose: To ensure that each function of the software operates as expected.
* Focus Area: User interface, APIs, database operations, user authentication, and overall application behavior.
* Example Scenarios:

1. Verifying that users can successfully log in with correct credentials.
2. Checking if a user can reset their password.
3. Ensuring a search feature returns the correct results.

* Tools Used: Selenium, JUnit, TestNG, QTP, etc.
* Type: Usually black-box testing, where the tester does not need to know the internal code.

Non-Functional Testing:

Non-functional testing focuses on how the system performs under certain conditions. It evaluates the application's quality attributes such as performance, usability, security, and scalability.

* Purpose: To ensure the system is reliable, efficient, and user-friendly under various conditions
* Focus Area: System speed, response time, resource usage, reliability, load handling, and user experience.
* Example Scenarios:

1. Checking how the system behaves with 1,000 simultaneous users.
2. Measuring the response time of a web page.
3. Verifying if the application is secure from unauthorized access.

* Tools Used: JMeter, LoadRunner, Postman (for performance APIs), Burp Suite (for security), etc.
* Type: Can be black-box or involve specialized tools and knowledge.

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

Ans. SDLC (Software Development Life Cycle):

Definition:  
SDLC is the process used by software developers to design, develop, test, and deliver high-quality software.

Focus:  
Covers the entire life cycle of software, from planning to deployment and maintenance.

Phases of SDLC:

1. Requirement Gathering
2. System Design
3. Implementation (Coding)
4. Testing
5. Deployment
6. Maintenance

Who is involved:  
Business analysts, designers, developers, testers, project managers, and stakeholders.

STLC (Software Testing Life Cycle)

Definition:  
STLC is a subset of SDLC that deals specifically with the testing phase of software development.

Focus:  
Ensures that the software is tested properly and meets the quality standards.

Phases of STLC:

1. Requirement Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup
5. Test Execution
6. Test Cycle Closure

Who is involved:  
Testers, test leads, QA managers.

Q 32. What isthe difference between test scenarios, test cases, and test script?

Ans. 1. Test Scenario

Definition:  
A test scenario is a high-level description of what to test. It represents a feature or functionality that needs to be tested.

Purpose:  
To ensure complete test coverage of the application.

2. Test Case

Definition:  
A test case is a detailed set of steps, conditions, inputs, and expected results to test a specific part of the application.

Purpose:  
To guide testers in executing the test and verifying expected behaviour.

3. Test Script

Definition:  
A test script is a set of instructions written in code or a tool (like Selenium, UFT) to automate the execution of a test case.

Purpose:  
To execute tests automatically and improve testing efficiency.

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

Ans. A Test Plan is a formal document that outlines the strategy, scope, objectives, resources, schedule, and activities involved in the software testing process.

It acts as a blueprint that guides the testing team through the testing lifecycle and ensures everyone understands what to test, how to test, when to test, and who will test.

**Key Information Covered in a Test Plan:**

1. Test Plan ID
2. Introduction / Overview
3. Objectives of Testing
4. Scope of Testing
   * In-Scope
   * Out-of-Scope
5. Test Strategy / Approach
6. Test Types to be Performed  
   (e.g., Functional, Regression, Smoke, Sanity, Performance)
7. Entry Criteria
8. Exit Criteria
9. Test Deliverables
10. Test Schedule / Timeline
11. Roles and Responsibilities
12. Test Environment Requirements
13. Test Tools / Automation Tools
14. Risks and Mitigation Plan
15. Defect Management Process
16. Approval / Sign-off

Q 34. What is priority?

Ans. In software testing, Priority refers to the urgency or importance of fixing a bug or implementing a feature from the business or customer perspective.

Q 35. What is severity?

Ans. In software testing, Severity refers to the impact of a defect on the application's functionality or system performance, regardless of how urgently it needs to be fixed.

Q 36. Bug categories are…

Ans. Bug categories are classifications of software defects based on their nature, origin, or impact. Here are some common bug categories:

1. Functional Bugs

* Description: Features or functionalities not working as intended.
* Examples:
  + A login button doesn’t respond.
  + Incorrect calculation in a payment system.

2. UI/UX Bugs

* Description: Issues related to user interface or experience.
* Examples:
  + Misaligned buttons.
  + Poor color contrast affecting readability.

3. Performance Bugs

* Description: Issues causing slow or inefficient system behavior.
* Examples:
  + Long page load times.
  + High CPU/memory usage.

4. Compatibility Bugs

* Description: Issues occurring only in specific environments.
* Examples:
  + Website breaks on certain browsers.
  + App crashes on a particular OS version.

5. Security Bugs

* Description: Vulnerabilities exposing the system to threats.
* Examples:
  + SQL injection flaws.
  + Weak password encryption.

6. Regression Bugs

* Description: Previously fixed bugs that reoccur after updates.
* Examples:
  + A feature that worked in v1.0 stops working in v1.1.

7. Integration Bugs

* Description: Issues arising from interactions between components/systems.
* Examples:
  + API failing to communicate with a third-party service.

8. Logic Bugs

* Description: Flaws in the code’s decision-making process.
* Examples:
  + A discount coupon applying incorrectly.

9. Boundary/Edge Case Bugs

* Description: Issues occurring at extreme input values.
* Examples:
  + App crashes when entering a very large number.

10. Workflow Bugs

* Description: Problems in multi-step processes.
* Examples:
  + Checkout process fails after selecting a payment method.

Q 37. Advantage of Bugzila

Ans.

1. Open-Source & Free – No licensing costs.
2. Highly Customizable – Adaptable workflows & fields.
3. Cross-Platform Support – Works on Windows, Linux, macOS.
4. Advanced Search & Filtering – Quick bug tracking with saved searches.
5. Detailed Reporting – Graphs, trends, and history logs.
6. Integration Support – Works with Git, SVN, APIs.
7. Role-Based Access Control – Secure user permissions.
8. Email Notifications – Real-time updates for team members.
9. Active Community Support – Open-source documentation & forums.
10. Scalability – Handles small to large projects efficiently.

Q 38. Difference between priority and severity.

Ans.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Priority** | **Severity** |
| Definition | How urgent it is to fix the bug | How serious the impact of the bug is |
| Focus | Business or customer urgency | Technical impact on the system |
| Decided By | Project Manager / Product Owner / Business Analyst | QA / Tester |
| Fix Timeline | Guides when it should be fixed | Indicates how badly it affects the system |
| Example (High) | Spelling mistake on home page of live website (high priority, low severity) | App crashes on login (high severity, high priority) |
| Example (Low) | Bug in a rarely used feature of the admin panel (low priority) | Minor UI glitch like misaligned button (low severity) |

Q 39. What are the different Methodologies in Agile Development Model?

Ans. **Popular Agile Methodologies:**

**1. Scrum**

* Work is divided into short iterations called Sprints (usually 1–4 weeks).
* Roles: Product Owner, Scrum Master, Development Team
* Events: Daily Stand-ups, Sprint Planning, Sprint Review, Retrospective
* Focuses on team collaboration, transparency, and continuous improvement.

**2. Kanban**

* Visualizes work using a Kanban board with columns like *To Do, In Progress, Done*.
* Focuses on continuous delivery and limiting work in progress (WIP).
* No fixed iterations — workflows continuously.
* Great for teams needing flexibility.

**3. Crystal**

* Emphasizes people and interactions over processes and tools.
* Comes in variants (Crystal Clear, Crystal Yellow, Crystal Red) depending on team size and project criticality.
* Adaptable and lightweight.

Q 40. Explain the difference between Authorization and Authentication in Web testing. What are the common problems faced in Web testing?

Ans.

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Authentication** | **Authorization** |
| Definition | Verifying who the user is | Determining what the user can access |
| Purpose | Confirms identity (login credentials) | Grants access to features, data, or resources |
| Process | User logs in with username/password | System checks if the user has permission |
| Example | Login with email and password | Accessing admin dashboard only if role = "Admin" |
| Occurs When? | Before authorization | After authentication |
| Common Tests | Login, password encryption, account lockout | Role-based access control, restricted URLs |

**The common problems faced in Web testing:**

1. Cross-browser compatibility issues
2. Responsive design problems on different devices
3. Slow website loading and performance issues
4. Broken links and navigation errors
5. Form validation failures
6. Security vulnerabilities (e.g., SQL injection, XSS)
7. Session management and cookie handling problems
8. Database errors and data inconsistency
9. Integration issues with third-party services/APIs
10. Incorrect or missing error messages
11. Content and UI layout problems
12. Inconsistent behavior in different environments (dev, staging, production)
13. Accessibility issues (not following standards for disabled users)
14. Caching problems leading to stale data display
15. File upload/download issues

Q 41. When to used Usablity Testing?

Ans. **The list of when to use Usability Testing:**

* During early design phases (wireframes/prototypes)
* Before final product release
* After major UI/UX changes
* When improving existing applications
* When targeting new user groups

Q 42. What is the procedure for GUI Testing?

Ans. **The procedure for GUI Testing:**

* Understand requirements
* Prepare test environment
* Create test cases
* Test layout and design
* Check functionality of UI elements
* Validate navigation flow
* Test responsiveness
* Check error handling and messages
* Perform cross-browser testing
* Execute accessibility testing
* Report defects
* Retest and regression testing