
QA
Software
Testing

Index

What is Software Testing?

The process checks whether the actual software matches the expected requirements and ensures the software is bug-free.

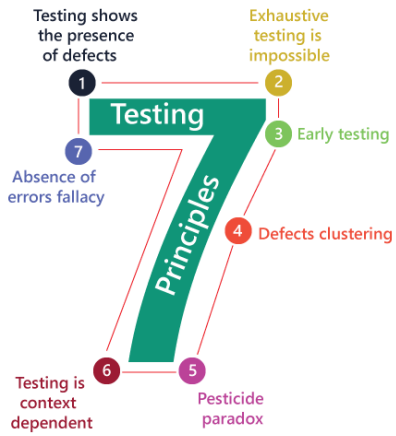
Verification

To ensure that the software correctly implements a specific function. It means “Are we building the product, right?”. [Correct Process]

Validation

To ensure that the software fulfils customer requirements. It means “Are we building the right product?”. [Correct Product]

7 Principles of Software Testing



1. **Testing shows the presence of defects:** Testing can reveal defects, but it cannot prove that the software is defect-free.
2. **Exhaustive testing is impossible:** It’s impractical to test all possible scenarios.
3. **Early testing:** Testing should start as early as possible in the software development lifecycle.
4. **Defect clustering:** A small number of modules usually contain most of the defects.
5. **Pesticide paradox:** Repeated tests become less effective at finding new defects over time.
6. **Testing is context-dependent:** Testing approaches vary based on the software’s context and requirements.
7. **Absence-of-errors fallacy:** A system free of known defects isn’t necessarily usable or fit for purpose.

Types of Software Testing

Based on Testing Level

- **Unit Testing:** Testing individual components or units of code (e.g., functions, methods).
- **Integration Testing:** Testing the interaction between integrated units or modules.

- **System Testing:** Testing the complete system to ensure it meets requirements.
- **Acceptance Testing:** Testing to ensure the software meets user requirements and is ready for delivery (includes UAT - User Acceptance Testing).

Based on Testing Approach

- **Manual Testing:** Tests are executed manually without automation tools.
- **Automated Testing:** Tests are executed using automation tools and scripts.
- **Black Box Testing:** Testing without knowledge of internal code structure; focuses on inputs and outputs.
- **White Box Testing:** Testing with knowledge of internal code structure; focuses on logic and paths.
- **Grey Box Testing:** A combination of black-box and white-box testing.

Based on Testing Purpose

- **Functional Testing:** Validates that the software functions as intended (e.g., unit, integration, system, and acceptance testing).
- **Non-Functional Testing:** Validates non-functional aspects like performance, usability, and reliability.
- **Performance Testing:** Tests speed, responsiveness, and stability under load.
- **Load Testing:** Tests behaviour under expected load conditions.
- **Stress Testing:** Tests behaviour under extreme conditions.
- **Usability Testing:** Tests user-friendliness and ease of use.
- **Security Testing:** Identifies vulnerabilities and ensures data protection.
- **Compatibility Testing:** Ensures the software works across different environments (e.g., browsers, devices).
- **Regression Testing:** Ensures new changes haven't broken existing functionality.

Based on Execution Timing

- **Static Testing:** Testing without executing the code (e.g., reviews, walkthroughs, inspections).
- **Dynamic Testing:** Testing by executing the code (e.g., functional and non-functional testing).

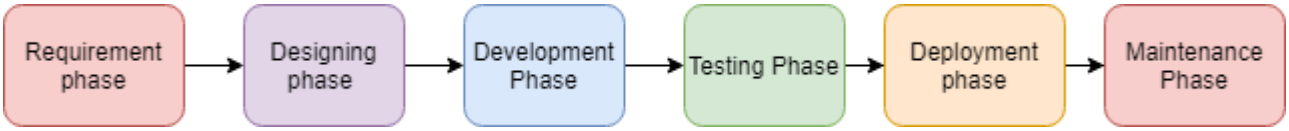
More Types

- **Smoke Testing:** A quick check to ensure basic functionality works.
- **Sanity Testing:** A narrow, focused test to ensure specific functionality works after changes.
- **Exploratory Testing:** Unscripted testing where testers explore the software dynamically.
- **Ad-hoc Testing:** Informal testing without planning or documentation.
- **Alpha Testing:** Testing performed by internal teams before release.
- **Beta Testing:** Testing performed by a limited group of end-users before final release.
- **Globalization Testing:** Testing app for multiple languages.

SDLC

- Structured process used by DEV teams to design, develop, test, and deploy software.
- It provides a framework for planning, controlling, and executing software projects.

SDLC Phases



SDLC Models

Different SDLC models define how these phases are executed. Common models include:

1. **Waterfall:** Sequential, linear approach.
2. **Agile:** Iterative and incremental development.
3. **Iterative:** Repeated cycles of development.
4. **Spiral:** Combines iterative development with risk analysis.
5. **V-Model:** Emphasizes testing at each development stage.
6. **DevOps:** Integrates development and operations for continuous delivery.

STLC

Structured process used by testing teams to ensure software quality through systematic testing.

1. Requirement Analysis

- Objective: Understand the testing requirements and scope.
- Key Activities:
    - Analyse functional and non-functional requirements.
    - Collaborate with stakeholders to clarify doubts.
    - Define the testing scope and objectives.

2. Test Planning

- Objective: Create a detailed test plan to guide the testing process.
- Key Activities:
    - Define testing goals, timelines, and resources.
    - Select testing tools and frameworks.
    - Identify risks and mitigation strategies.
    - Prepare the Test Plan document.

3. Test Case Development

- Objective: Create test cases and scripts for execution.
- Key Activities:
    - Write detailed test cases based on requirements.

- Review and approve test cases.
- Prepare test data and environment setup.
- Develop automation scripts (if applicable).

4. **Test Environment Setup**

- Objective: Prepare the environment for testing.
- Key Activities:
  - Set up hardware, software, and network configurations.
  - Deploy the application build.
  - Ensure the environment mirrors the production setup.

5. **Test Execution**

- Objective: Execute test cases and report defects.
- Key Activities:
  - Run test cases (manual or automated).
  - Log defects in a tracking tool (e.g., JIRA).
  - Retest fixed defects (regression testing).
  - Track test progress and report status.

6. **Test Cycle Closure**

- Objective: Evaluate the testing process and prepare reports.
- Key Activities:
  - Analyse test results and metrics.
  - Prepare a Test Closure Report.
  - Share feedback for process improvement.

Testing Techniques

1. **Error Guessing**

- Description: A technique where testers use their experience, intuition, and knowledge of the system to guess where errors or defects might occur.
- When to Use: When there is no formal requirement documentation or when time is limited.
- Example: Guessing that a login system might fail if special characters are used in the password field.

2. **Equivalence Partitioning**

- Description: Divides input data into partitions (equivalence classes) where the system is expected to behave similarly. Test one value from each partition.
- When to Use: To reduce the number of test cases while maintaining coverage.
- Example: For a field accepting values 1–100:

- Valid partition: 50 (any value between 1 and 100).
- Invalid partitions: -1 (below range) and 101 (above range).

3. Boundary Value Analysis (BVA)

- Description: Tests values at the edges of input ranges, as errors are expected at boundaries.
- When to Use: When input ranges are defined.
- Example: For a field accepting 1–100:
  - Test values: 0, 1, 100, 101.

4. **Decision Table Testing**

- Description: Tests combinations of inputs and their respective outputs using a table format.
- When to Use: When there are multiple inputs and complex business rules.
- Example: Testing a login system:

Username	Password	Output
Valid	Valid	Login Success
Valid	Invalid	Login Failed
Invalid	Valid	Login Failed
Invalid	Invalid	Login Failed

5. **All-Pairs Testing (Pairwise Testing)**

- Description: Tests all possible discrete combinations of input parameters to ensure coverage of interactions between parameters.
- When to Use: When there are multiple input parameters, and exhaustive testing is impractical.
- Example: Testing a form with three fields (e.g., name, age, gender) by pairing values:
    - Test Case 1: Name A, Age 20, Gender Male
    - Test Case 2: Name A, Age 30, Gender Female
    - Test Case 3: Name B, Age 20, Gender Female
    - Test Case 4: Name B, Age 30, Gender Male

6. **Cause-Effect Testing**

- Description: Identifies causes (inputs) and effects (outputs) and creates test cases to validate the relationships between them.
- When to Use: When there are clear cause-effect relationships in the system.
- Example: Testing an ATM withdrawal:
    - Cause: Valid card + Correct PIN + Sufficient balance.
    - Effect: Cash dispensed.

7. **State Transition Testing**

- Description: Tests system's behaviour as it transitions btw states based on inputs or events.
- When to Use: When the system has defined states and transitions (e.g., login systems, workflows).
- Example: Testing a login system:
    - State 1: Initial state (no input).
    - State 2: After entering a valid username.
    - State 3: After entering a valid password.
    - State 4: Successful login.

8. **Use Case Testing**

- Description: Tests end-to-end user scenarios or workflows based on use cases.

- When to Use: To validate real-world user interactions with the system.
- Example: Testing an e-commerce checkout process:
    - Use Case: User adds items to the cart, proceeds to checkout, enters payment details, and completes the purchase.

Types of Integration Testing

1. **Incremental Testing**
   Involves testing modules step-by-step by adding and verifying them gradually.

- Top-Down Approach – Testing starts from the topmost module and moves downward.
    - Example: Testing a banking app where the UI is tested first, followed by API and database.
    - Use When: The high-level modules are ready before lower-level modules.
- Bottom-Up Approach – Testing starts from lower-level modules and moves upward.
    - Example: Testing a payment gateway where APIs are tested first, followed by UI.
    - Use When: The core functionalities (backend) are stable before the UI.

2. **Non-Incremental Testing**
   All modules are integrated and tested together in one go.
    - Example: A social media app where all features like posts, notifications are tested at once.
    - Use When: The system is small, or all modules are available simultaneously.

Test Scenario

- A high-level description of what needs to be tested. Ensures that all user interactions and workflows are covered during testing.
- Characteristics:
    - Derived from requirements or use cases.
    - Focuses on "what to test" rather than "how to test."
    - Usually, one-liner statements or brief descriptions.

Test Cases

- A detailed set of steps, conditions, and inputs used to verify a specific functionality.
- Derived from a Test Scenario. To validate a specific func. works as expected.
- Characteristics:
    - Detailed and step-by-step.
    - Includes preconditions, test steps, expected results, and postconditions.
    - Focuses on "how to test."

Test Documentation

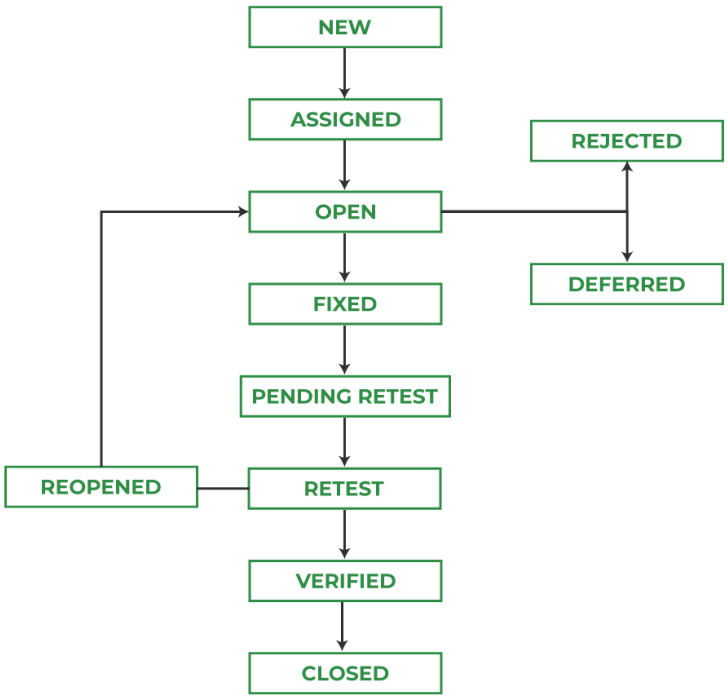
1. **Test Plan** – Outlines testing scope, objectives, schedule, resources, and strategy.
2. **Test Strategy** – High-level doc defining the overall testing approach and methodologies.
3. **Test Scenario** – High-level test idea covering various user actions to validate functionality.
4. **Test Case** – Detailed step-by-step procedure to verify specific func with expected results.
5. **Test Script** – Automated test cases written in a programming language for test execution.
6. **Test Data** – Input values required for executing test cases.
7. **Traceability Matrix (RTM)** – Maps requirements to test cases to ensure complete coverage.
8. **Test Execution Report** – Summarizes test case execution status and defects identified.
9. **Defect Report** – Logs details of defects found, including severity, priority, and steps to reproduce.
10. **Test Summary Report** – Final report summarizing testing activities, results, and overall product quality.

Bug

It is the informal name of defects, means software or app is not working as per the requirement.

Terms	Description	Raised by
Defect	When the application is not working as per the requirement.	Test Engineer
Bug	Informal name of defect	Test Engineer
Error	Problem in code leads to the errors.	Developer, Automation Test Engineer
Issue	When the application is not meeting the business requirement.	Customer
Mistake	Problem in the document is known as a mistake.	--
Failure	Lots of defect leads to failure of the software.	--

Bug Life cycle



Severity

- The impact of the bug on the application is known as severity.
- It can be a **blocker, critical, major, and minor** for the bug.

Priority

- Priority is importance for fixing the bug or which bug to be fixed first or how soon the bug should be fixed.
- It can be **urgent, high, medium, and low**.

What is Build?

- In manual testing, build is software that contains some set of features/bugs, and it is installed on a test server that needs to be tested for the product's stability.
- Every new build will be the improved version of the new build.

What is Release?

- The Release is a final product or a project, which is delivered to the customer.
- Entirely developed application, while the build is the part of an application or the software.

Questions

1. What is a User Story?
    a. Description of a software feature from an end-user perspective.

2. What is Dynamic Testing?
    a. Dynamic testing is a software testing technique where the dynamic behaviour of the code is checked.

3. What is Risk-Based Testing?
    a. It uses risks to prioritize the appropriate tests during test execution.
        i. Software complexity.
        ii. Criticality of the businesses.
        iii. Frequency of use.
        iv. Possible areas of defects.

4. What is Defect Cascading?
    a. Defect Cascading is when one defect leads to the discovery of another defect by software testers.
    b. This could occur when original defect was not fixed.

5. What is Test-Driven Development?
    a. iterative approach that involves combining the two processes, the creation of test cases, and refactoring.
    b. test cases are created for each functionality and tested first and if the test fails then a new code is written to pass the test, thus making the code bug-free.

6. What parameters help check the quality of test execution?
    a. The **defect rejection ratio** is the ratio of total rejections to total Production.
    b. The **defect leakage ratio** is the ratio of total rejections to total Production.

7. What are Latent Defects?
    a. Latent defect is a term used to identify hidden undetected flaws in software using some special operations set. These defects only occur when software testing triggers a specific event, concealing their presence.

8. What is Mutation Testing?
    a. To validate a set of test data or test cases is useful. It is done by deliberately adding multiple code changes (bugs) and retesting with the original test cases and data.

9. What is cross-browser Testing

   a. Check whether your website or app works as intended when using different Browser-OS combinations. Like Chrome, Mozilla Firefox, Microsoft Edge, Safari, and more.

10. What are exit criteria in Software Testing?
    a. Specific conditions which specify the agreed-upon features or state of the application to indicate the completion of the product or process.

11. What basis can you arrive at for an estimation for your project?
    a. To estimate any project, you need to consider the following points:
        i. Divide the entire project into the minor tasks
        ii. Allocate each task to team members
        iii. It helps you to estimate the effort required to complete each task
        iv. Validate the estimation

12. What is Error Seeding?
    a. It is a method of adding known errors in a program that helps you identify the error detection rate. It helps in estimating the tester's skills of finding bugs.

13. What do you mean by Defect Triage?
    a. helps you find defects prioritized depending on different characteristics like risk, severity and the amount of time it should take to fix the problem.
    b. Various stakeholders are involved, such as the development team, testing team, project manager, etc.

14. What is Pesticide Paradox?
    a. When repetitive test cases do not reveal new bugs.
    b. TCs needs regular add or update to find more defects.

15. What is the difference between the QA and software testing?
    a. QA means to monitor the quality of the "process" used to produce the software.
    b. Software testing is the process of ensuring the functionality of final product meets the user's requirement.

Differences

Smoke Testing	Sanity Testing
To verify the stability of the build.	To ensure no new issues are introduced.
Broad (covers major functionalities).	Narrow (focuses on specific areas).
Performed after a new build or deployment.	Performed after minor changes or bug fixes.
Documented	Not Documented

Regression Testing	Retesting
Ensures new changes don't break existing func	Verifies fixed defects.
Performed on unchanged parts of application.	Performed on the earlier failed tests only.
Automated in most cases.	Always manual.

Static Testing	Dynamic Testing
Performed without executing code.	Performed after executing code.
Focuses on documentation and code review.	Focuses on fucn and n-func behaviour.
Ex: Walkthrough, Inspections	Ex: System, unit & integration

Load Testing	Stress Testing
Test the system under expected load.	Test system beyond capacity.
Focuses on performance and response time.	Focuses on system stability and recovery.
Ex: Simulating 1000 users.	Ex: Simulating 5k users on a sys designed for 1k

Boundary Value Analysis	Equivalence Partitioning
Focuses values at the edges of input range.	Divides input into valid and invalid partitions.
Ex: Testing minimum and maximum values.	Tests one value from each partition.
Identifies off-by-one errors.	Reduces the number of test cases.

1 Real-Life Scenario-Based Questions (Lift, ATM, E-commerce)

1. **How would you test a lift (elevator)?**

✅ Check floor selection, emergency button, overload, door closing timing, power failure handling.

2. **How would you test an ATM machine?**

✅ Validate card insertion, PIN verification, cash withdrawal, receipt printing, insufficient balance.

3. **How would you test a vending machine?**

✅ Test payment options, product selection, out-of-stock handling, refund scenarios.

4. **How would you test a fuel dispenser at a petrol station?**

✅ Validate fuel selection, quantity accuracy, receipt generation, emergency stop.

5. **How would you test a digital clock?**

✅ Check time format, alarm function, daylight savings adjustments, battery backup.

6. **How would you test an escalator?**

✅ Verify step movement, emergency stop, load handling, sensor detection.

7. **How would you test an online ticket booking system?**

✅ Validate search filters, seat selection, payment handling, cancellation, and refund.

8. **How would you test a voice recognition system?**

✅ Validate different accents, background noise impact, voice matching accuracy.

9. **How would you test an online shopping cart?**

✅ Add/remove products, quantity change, checkout process, discount application.

10. **How would you test a barcode scanner?**

✅ Scan different barcodes, invalid barcode handling, speed and accuracy.

2 Logical Reasoning & Testing IQ Questions

11. **A website loads slowly only on mobile but works fine on desktop. Identify the issue?**
    ✅ Mobile network speed, image compression, responsive design issues.
12. **A calculator shows the wrong sum for large numbers. What could be wrong?**
    ✅ Integer overflow issue, wrong data type used.
13. **A user reports that clicking a button does nothing. How will you debug?**
    ✅ Check event binding, inspect console errors, verify network requests.
14. **A login page crashes only when a user enters 100+ characters. Why?**
    ✅ Possible buffer overflow or unhandled input validation.
15. **A dropdown menu is not opening on a webpage. What will you check?**
    ✅ Verify JavaScript errors, inspect CSS styles, check browser compatibility.
16. **A webpage displays '404 Not Found' after deployment. What could be wrong?**
    ✅ Broken URL, incorrect routing, missing files on the server.

17. **A form submission works in Chrome but fails in Firefox. What will you check?**
    ✅ Cross-browser compatibility, JavaScript errors, API request handling.
18. **A website takes too long to load after adding new images. How will you fix it?**
    ✅ Optimize images, use caching, implement lazy loading.
19. **A mobile app crashes when switching from Wi-Fi to mobile data. Why?**
    ✅ Network handling issue, improper API retries.
20. **A user reports that they can't reset their password. How do you investigate?**
    ✅ Check email delivery, link expiration, password policy enforcement.

3️⃣ Edge Case & Negative Testing Scenarios

21. **What happens if a user enters a negative value in an age field?**
    ✅ System should reject it with an error message.
22. **What if a user enters emojis in the name field?**
    ✅ Verify if system allows or rejects it.
23. **What if the power goes off while transferring money online?**
    ✅ System should either complete or roll back the transaction.
24. **What happens if an email is sent without an '@' symbol?**
    ✅ System should prompt an invalid email format error.
25. **What if a user submits a form without filling any fields?**
    ✅ System should show validation messages.
26. **What if a user uploads a virus-infected file in a file upload feature?**
    ✅ System should detect and reject it.
27. **What happens if 100 users submit a form at the same time?**
    ✅ Test for concurrency handling and server load balancing.
28. **What if a website runs out of stock while a user is making a purchase?**
    ✅ System should notify the user and prevent checkout.
29. **What if a user's internet disconnects during payment?**
    ✅ Ensure payment retry or rollback mechanism.
30. **What happens if two users try to book the same seat at the same time?**
    ✅ System should allow only one booking and notify the other user.

4 Functional Testing Scenarios

31. **How will you test a password reset feature?**
    ✅ Check valid/invalid email, token expiration, and reset functionality.
32. **How will you test a search functionality?**
    ✅ Test with valid keywords, empty search, incorrect spelling, and SQL injection.
33. **How will you test a contact form?**
    ✅ Validate required fields, input length limits, and email format.
34. **How will you test a video streaming website?**
    ✅ Test buffering, playback speed, resolution switching, and subtitle synchronization.
35. **How will you test a GPS tracking app?**
    ✅ Verify live location updates, accuracy, and route recalculations.

5 Performance & Security Testing Scenarios

36. **How would you test website load time?**
    ✅ Use tools like GTmetrix, Lighthouse, or JMeter.
37. **How would you test API response time?**
    ✅ Use Postman or JMeter to analyze latency and performance.
38. **What security tests will you perform on a login page?**
    ✅ Test SQL injection, brute force attack, session hijacking, and password encryption.
39. **What happens if a website has no HTTPS?**
    ✅ User data is not encrypted, increasing security risks.
40. **How do you test CAPTCHA functionality?**
    ✅ Check if bots can bypass it, ensure it refreshes properly.

6 Database & Backend Testing Scenarios

41. **How would you verify database integrity?**
    ✅ Check data consistency after CRUD operations.
42. **What if an API call returns a 500 error?**
    ✅ Check server logs, request payload, and error handling.
43. **How would you test data migration?**
    ✅ Validate data before and after migration for correctness.
44. **How do you test session timeouts?**
    ✅ Stay idle and verify auto-logout behaviour.

45. **How do you test a database backup process?**
    - ✅ Verify successful backup creation and restore functionality.

7️⃣ Miscellaneous Tricky Questions

46. **Why do testers need exploratory testing?**
    - ✅ To find unexpected defects and edge cases.
47. **What if a test case fails but the application still works fine?**
    - ✅ Review expected vs. actual results before logging a defect.
48. **What is the difference between bug leakage and bug release?**
    - ✅ Bug leakage: Found after release. Bug release: Known bug accepted for release.
49. **Why do we need both manual and automated testing?**
    - ✅ Manual testing covers exploratory testing; automation speeds up regression.
50. **How do you prioritize test cases when there is a time constraint?**
    - ✅ Focus on high-risk areas and critical functionalities first.