

100 Most Asked QA Interview Questions

Manual testing · Agile · SQL · Selenium · Playwright · API · CI/CD · AI testing

JS Testing Academy

jstestingacademy.co.in

+91 73580 36583 · info@jstestingacademy.co.in

Manual Testing & QA Fundamentals

1. What is the difference between verification and validation?

Verification asks 'are we building the product right?' – reviews, walkthroughs, static checks. Validation asks 'are we building the right product?' – actual execution against requirements.

2. What is the difference between severity and priority?

Severity is the technical impact of the defect; priority is how soon the business wants it fixed. A spelling mistake on the homepage logo is low severity but high priority; a crash in a rarely used admin report is high severity, low priority.

3. Explain the defect life cycle.

New → Assigned → Open/In progress → Fixed → Retest → Closed. Alternative paths: Rejected, Duplicate, Deferred, Reopened (when a retest fails).

4. What is the difference between smoke and sanity testing?

Smoke is a shallow, wide check that the build is stable enough to test. Sanity is a narrow, deeper check of a specific fix or feature after a minor change.

5. What is the difference between retesting and regression testing?

Retesting verifies that a specific fixed defect is actually fixed, using the same data. Regression checks that the fix did not break anything else.

6. What are the levels of testing?

Unit, integration, system and acceptance testing (UAT).

7. What is the STLC?

Requirement analysis, test planning, test case design, environment setup, test execution, and test closure – each with entry and exit criteria.

8. What is equivalence partitioning?

Dividing input data into groups that should behave the same way and testing one value per group – e.g. age 18–60 valid, below 18 invalid, above 60 invalid.

9. What is boundary value analysis?

Testing at the edges of a range because defects cluster there: for 18–60, test 17, 18, 19, 59, 60, 61.

10. What is a decision table and when do you use it?

A table mapping combinations of conditions to expected actions. Use it when business rules depend on several conditions together, such as discount rules.

11. What is state transition testing?

Testing how the system moves between states – for example an order moving from Created → Paid → Shipped → Delivered, including invalid transitions.

12. What is exploratory testing?

Simultaneous learning, test design and execution, guided by a charter and time-boxed sessions. Notes and findings are recorded as you go.

13. When do you stop testing?

When exit criteria are met: planned coverage executed, no open blocker/critical defects, defect arrival rate falling, deadlines and risk accepted by stakeholders.

14. What is a test scenario versus a test case?

A scenario is a one-line business condition to be tested; a test case is the detailed steps, data and expected result to verify it.

15. What would you do if there are no requirements?

Use the application, existing tickets and production data to build an implied spec, talk to BA/support/users, test against standards and competitor behaviour, and document every assumption for review.

16. What is risk-based testing?

Prioritising what you test by business impact and probability of failure, so limited time is spent where defects hurt most.

17. What is the difference between static and dynamic testing?

Static testing reviews artefacts without executing code (reviews, walkthroughs, static analysis).
Dynamic testing executes the application.

18. What is a Requirement Traceability Matrix?

A mapping of requirements to scenarios, test cases, execution status and defects – it proves coverage in both directions.

19. What are entry and exit criteria?

Conditions that must be met to start a test phase (environment ready, build deployed, cases reviewed) and to complete it (coverage achieved, defects closed or accepted).

20. What is UAT and who performs it?

User Acceptance Testing – business users validate real-world scenarios against business needs in a production-like environment before go-live.

21. How do you write a good bug report?

A precise title, environment and build, numbered reproduction steps, expected vs actual, severity and priority, plus evidence (screenshot, video, logs) and test data.

22. What is the difference between a defect, error, bug and failure?

An error is a human mistake, which introduces a defect (bug) in the code; a failure is what the user sees when that defect is executed.

23. What is shift-left testing?

Starting testing activities earlier – requirement reviews, acceptance criteria, unit and API tests – so defects are found before the UI exists.

24. What is a test closure report?

A summary of objectives vs achievement, execution and defect statistics, deferred defects, lessons learned and reusable assets handed over.

25. What is accessibility testing?

Verifying the product can be used by people with disabilities – keyboard access, screen reader support, colour contrast, labels and focus order (WCAG).

Agile & Process

26. What are the Scrum ceremonies?

Sprint planning, daily stand-up, sprint review, sprint retrospective, plus ongoing backlog refinement.

27. What is the definition of done for a story?

An agreed checklist – code reviewed, unit tests passing, acceptance criteria met, tests automated, documentation updated, deployed to QA and verified.

28. How do you fit regression into a two-week sprint?

Automate the stable regression set and run it nightly in CI; manually re-test only the areas touched by this sprint's changes, prioritised by risk.

29. What do you do if a story has no acceptance criteria?

Do not start. Raise it in refinement, propose criteria yourself from the discussion, and get the PO to confirm before the story enters the sprint.

30. What is a spike in Agile?

A time-boxed investigation to reduce uncertainty – e.g. to evaluate a tool or approach – producing knowledge rather than shippable features.

31. How do you estimate testing effort?

Break work into tasks, use historical velocity or three-point estimates, add time for environment, data, retesting and regression, and state assumptions.

32. What QA metrics do you report?

Execution progress, defect density, defect ageing, escaped defects, automation coverage and pass rate, regression cycle time.

33. What is escaped defect leakage?

Defects found in production that testing missed, usually expressed as a ratio against total defects – it drives process improvement.

SQL & Database Testing

34. What is the difference between WHERE and HAVING?

WHERE filters rows before grouping; HAVING filters groups after GROUP BY aggregation.

35. Write a query to find the second highest salary.

SELECT MAX(salary) FROM employees WHERE salary < (SELECT MAX(salary) FROM employees); or use DENSE_RANK() in a subquery and filter rank = 2.

36. How do you find duplicate records?

SELECT email, COUNT(*) FROM users GROUP BY email HAVING COUNT(*) > 1;

37. What is the difference between INNER JOIN and LEFT JOIN?

INNER JOIN returns only matching rows from both tables; LEFT JOIN returns all rows from the left table with NULLs where the right has no match.

38. What is the difference between DELETE, TRUNCATE and DROP?

DELETE removes selected rows and can be rolled back; TRUNCATE removes all rows quickly and resets identity; DROP removes the table structure itself.

39. What is a primary key versus a foreign key?

A primary key uniquely identifies a row and cannot be NULL; a foreign key references a primary key in another table and enforces referential integrity.

40. What is an index and why does it matter in testing?

An index speeds up reads at the cost of writes and storage. In testing it matters for performance scenarios and for explaining slow queries.

41. How do you validate data after an API call?

Query the database for the record, check every field written, verify audit columns and relationships, and confirm no duplicate or orphan rows were created.

42. What is the difference between UNION and UNION ALL?

UNION removes duplicate rows and sorts; UNION ALL returns everything including duplicates and is faster.

43. How do you test an ETL / data migration?

Compare source and target counts, reconcile key totals, validate transformation rules row by row on a sample, check data types, nulls, duplicates and rejected records.

Automation & Selenium

44. Why do we automate, and what should not be automated?

Automate stable, repetitive, high-risk and data-driven checks. Do not automate one-off tests, unstable features, exploratory or usability testing.

45. What locator strategy do you use in Selenium?

id or data-testid first, then name, then CSS, then relative XPath. Avoid absolute XPath and text that changes with locale.

46. What is the difference between implicit, explicit and fluent waits?

Implicit sets a global polling timeout for element lookup; explicit waits for a specific condition on a specific element; fluent adds polling interval and ignored exceptions.

47. Why should you avoid Thread.sleep?

It always waits the full time even if the element is ready, making suites slow, and it still fails when the app is slower than the fixed wait.

48. What is StaleElementReferenceException and how do you handle it?

The element reference is no longer attached to the DOM after a re-render. Re-find the element, use page object methods that locate on use, or retry with an explicit wait.

49. How do you handle dropdowns, alerts, frames and windows?

Select class for <select> dropdowns, driver.switchTo().alert() for alerts, switchTo().frame() for iframes, and window handles for new tabs or windows.

50. What is the Page Object Model and why use it?

Each page is a class exposing actions; locators live in one place. Tests read as business steps and UI changes are fixed in a single file.

51. How do you run tests in parallel?

Use the runner's parallel support (TestNG parallel attribute, pytest-xdist, Playwright workers), keep tests independent and thread-safe, and isolate test data.

52. How do you handle test data in automation?

External files or a data factory, created fresh per test where possible, cleaned up after, with no dependency on data that another test may change.

53. How do you handle file upload and download?

Upload by sending the file path to the input element; for downloads set the browser's download directory in options and verify the file on disk.

54. What is Selenium Grid?

A hub-and-node setup that runs tests on different browsers and machines, usually in Docker containers, for parallel and cross-browser execution.

55. How do you take a screenshot on failure?

Hook into the test listener (TestNG ITestListener, pytest hook, Playwright trace/screenshot options) and capture the screenshot in the failure handler.

56. What makes an automation suite flaky, and how do you fix it?

Hard waits, shared state, order dependency, dynamic locators, animations and network timing. Fix with stable test ids, explicit waits, isolated data and retries only as a last resort.

57. How do you decide UI versus API test coverage?

Put business logic, validation and data checks at the API level; keep UI tests for user journeys and rendering. It is faster, more stable and cheaper to maintain.

58. What reporting do you use?

Allure, Extent or the framework's HTML report, with screenshots, logs and traces attached to failures, published as a CI artefact.

Playwright, Cypress & Modern Tools

59. How does Playwright differ from Selenium?

Playwright has built-in auto-waiting, browser contexts for isolation, network interception, trace viewer and a bundled test runner; Selenium is a W3C driver library with a wider language and browser ecosystem.

60. What is auto-waiting in Playwright?

Before acting, Playwright waits for the element to be attached, visible, stable, enabled and able to receive events – removing most explicit waits.

61. What is storageState used for?

Saving cookies and local storage after a login so other tests reuse the session instead of logging in again.

62. What are fixtures in Playwright Test?

Reusable setup and teardown injected into tests – page, context, custom fixtures such as a logged-in page or test data.

63. What is the trace viewer?

A recorded timeline of the test with DOM snapshots, actions, network and console output – the fastest way to debug a CI failure.

64. How do you test APIs inside Playwright?

Use the request fixture / APIRequestContext to call endpoints directly, for setup, assertions or full API tests in the same suite.

65. How does Cypress differ from Selenium and Playwright?

Cypress runs inside the browser event loop with time-travel debugging and easy stubbing, but is JavaScript-only with limited multi-tab and cross-origin support.

66. How do you achieve cross-browser testing?

Projects/capabilities for Chromium, Firefox and WebKit/Edge, run in CI matrices or on a cloud device farm such as BrowserStack or Sauce Labs.

67. What is visual regression testing?

Comparing screenshots against approved baselines to catch layout and styling changes, with tolerance thresholds and masked dynamic regions.

68. How do you keep an automation suite under 10 minutes?

Shift checks to the API layer, run in parallel shards, reuse authentication state, avoid sleeps, and split smoke from full regression.

API Testing

69. What is the difference between REST and SOAP?

REST is an architectural style using HTTP verbs and usually JSON; SOAP is a protocol with XML envelopes, WSDL contracts and built-in standards for security and transactions.

70. What do 200, 201, 204, 400, 401, 403, 404, 409, 422, 429 and 500 mean?

Success, created, no content; bad request, unauthenticated, forbidden, not found, conflict, validation failed, rate limited; and a server error.

71. What is the difference between PUT and PATCH?

PUT replaces the whole resource and is idempotent; PATCH applies a partial update.

72. What is idempotency and why does it matter?

The same request repeated gives the same result. It matters for retries – a duplicate payment request must not charge twice.

73. How do you test an API without documentation?

Capture traffic from the UI with a proxy, inspect requests and responses, build a collection, then probe required fields, types, auth and error behaviour.

74. What do you validate in an API response?

Status code, response time, headers, schema and data types, field values, pagination, error format, and the resulting database state.

75. How do you handle authentication in API tests?

Fetch a token once in setup (OAuth 2.0 client credentials or login), store it in a request spec or fixture, and test expired, missing and wrong-role tokens explicitly.

76. What is JSON schema validation and why use it?

It checks the response structure and types against a contract, catching backend changes that field-by-field assertions would miss.

77. What is the difference between path, query and body parameters?

Path identifies the resource (/users/2), query filters or paginates (?page=2), body carries the payload for POST/PUT/PATCH.

78. How do you test rate limiting?

Send requests above the configured limit, expect 429 with a Retry-After header, confirm normal service resumes after the window and that limits are per user or per key as designed.

79. What are negative API test cases?

Missing or null required fields, wrong types, over-long values, invalid enum values, malformed JSON, wrong content type, unauthorised access and non-existent ids.

80. What is contract testing?

Verifying that the provider's API still satisfies the consumer's expected contract, so services can be released independently.

CI/CD, DevOps & Frameworks

81. What is CI/CD and where do tests fit?

Continuous integration builds and tests every commit; continuous delivery/deployment ships it. Unit and API tests run per commit, smoke after deploy, full regression nightly.

82. What is a quality gate?

An automated threshold in the pipeline – test pass rate, coverage, critical defects, security findings – that must be met before the build can progress.

83. How do you run tests in Docker?

Containerise the suite and browsers (Selenium Grid images or Playwright image), run with docker compose in CI, and publish reports as artefacts.

84. What is the test pyramid?

Many fast unit tests, fewer API/integration tests, few UI end-to-end tests – the opposite shape (the ice-cream cone) is slow and flaky.

85. What should a good framework contain?

Config management, page/service layer, independent tests, external test data, logging, reporting, parallel execution, CI integration and a README that lets a new joiner run it.

86. How do you handle different environments in a framework?

A config file or environment variables per environment, chosen at runtime (-Denv=qa), with no hard-coded URLs or credentials in code.

87. How do you store credentials used by tests?

In CI secrets or a vault, injected as environment variables – never committed to the repository.

88. What is your Git branching approach for test code?

Feature branches with pull requests and review, trunk-based merges, tests running on the PR before merge.

89. How do you deal with a red build every morning?

Triage first: product bug, test bug or environment. Quarantine the flaky tests, fix root causes, and keep an owner for the pipeline so red never becomes normal.

90. What is shift-right testing?

Validating in production – monitoring, synthetic checks, canary releases, feature flags and real-user telemetry.

AI in Testing

91. How do you use AI in your daily testing work?

Drafting test scenarios from acceptance criteria, generating test data, converting manual cases to automation skeletons, reviewing code for weak locators, and summarising failures – always with human review.

92. What are the risks of AI-generated test cases?

They look plausible but can miss product-specific rules, invent requirements, duplicate coverage or drop negative cases. Review and map them to acceptance criteria.

93. What is self-healing automation and what is the risk?

The framework re-identifies an element when its locator changes. The risk is masking a genuine UI change – so healed steps must be logged and reviewed.

94. How do you test an LLM-based feature?

Build an evaluation dataset with expected behaviour, score accuracy, check refusals, hallucination, prompt injection, data leakage, format compliance and re-run after every prompt or model change.

95. What is RAG and how do you test it?

Retrieval-Augmented Generation retrieves context from a knowledge base before generating. Test retrieval relevance, grounding of the answer, citations, and behaviour when nothing relevant is found.

96. What is prompt injection and how do you test for it?

Malicious instructions inside user input or documents that try to override the system prompt. Test with phrases like 'ignore previous instructions', hidden text in uploads, and verify the guardrails hold.

97. How do you assert on non-deterministic output?

Assert on structure and properties – valid JSON, required fields, no banned content, semantic similarity against a reference, or an LLM-as-judge score with a threshold – not exact string equality.

98. Can AI replace testers?

No. It accelerates drafting and analysis, but risk assessment, business context, judgement about what matters and accountability for release decisions stay with the tester.

99. What data should never be pasted into a public AI tool?

Customer personal data, credentials, API keys, production exports and anything covered by an NDA or compliance rule.

100. How do you measure whether AI actually helped?

Compare before and after: maintenance hours, time to author a suite, regression cycle time, escaped defects and review effort.

Want detailed answers, topic-wise question banks, MCQ practice tests and mock interviews?
Visit jstestingacademy.co.in → Resources → Interview Preparation. © JS Testing Academy