

SOFTWARE TESTING & MANUAL TESTING

Interview Questions & Answers

Basic • Intermediate • Tricky • Scenario-Based • Senior-Level

141 Interview Questions with Answers

A practical preparation guide for Software Tester, QA Engineer, Manual Tester and Senior QA interviews.

ROTEBIT

<https://rotebit.com>

Contents

Software Testing Fundamentals — 15 questions

SDLC, STLC and Test Process — 13 questions

Testing Types and Levels — 18 questions

Test Design Techniques — 12 questions

Positive, Negative and Edge Cases — 8 questions

Defects and Tricky Questions — 15 questions

Web and UI Manual Testing — 10 questions

API and Backend Manual Testing — 13 questions

Database and Data Validation — 9 questions

Agile and Scrum — 8 questions

Scenario-Based and Senior Questions — 20 questions

How to use this guide: Learn the principle, then practice explaining it in your own words and add a project example. For tricky questions, state assumptions before answering.

Software Testing Fundamentals

1. What is software testing?

Answer: Testing is the systematic evaluation of software to find defects, verify requirements, and assess whether the product is fit for intended use.

2. What is Quality Assurance (QA)?

Answer: QA is a broader quality discipline focused on improving processes and preventing defects, while testing is primarily concerned with evaluating the product.

3. What is verification?

Answer: Verification asks whether we are building the product right. It includes reviews, inspections, walkthroughs, and other static activities.

4. What is validation?

Answer: Validation asks whether we are building the right product. It evaluates the actual software against user and business needs.

5. What is a test case?

Answer: A documented set of preconditions, inputs, actions, and expected results used to verify a specific behavior.

6. What is a test scenario?

Answer: A high-level statement of what should be tested. One scenario can result in several detailed test cases.

7. What is a test condition?

Answer: A testable aspect of a requirement, feature, transaction, quality attribute, or risk.

8. What is a defect?

Answer: A flaw in a work product that can cause the software to fail to meet a requirement or expected behavior.

9. Error vs defect vs failure?

Answer: An error is a human mistake; a defect is the resulting flaw; a failure occurs when execution produces an incorrect result or behavior.

10. What is a test oracle?

Answer: A source or mechanism used to determine whether an actual result is correct, such as a requirement, specification, business rule, or invariant.

11. What is test data?

Answer: Data prepared or selected to exercise particular test conditions and expected outcomes.

12. What makes a good test case?

Answer: It should be clear, traceable, repeatable, maintainable, and contain an unambiguous expected result.

13. What is traceability?

Answer: Traceability links requirements to tests and defects so coverage, impact, and missing validation can be assessed.

14. What is a test suite?

Answer: A collection of related test cases grouped for execution.

15. What is a test environment?

Answer: The hardware, software, network, configuration, services, and data used to execute tests.

SDLC, STLC and Test Process

16. What is SDLC?

Answer: Software Development Life Cycle is the overall process used to plan, build, test, release, and maintain software.

17. What is STLC?

Answer: Software Testing Life Cycle covers test analysis, planning, design, environment preparation, execution, defect reporting, closure, and lessons learned.

18. What happens during requirement analysis?

Answer: Testers identify testable requirements, ambiguities, dependencies, risks, acceptance criteria, and missing information.

19. What is a test plan?

Answer: A document defining test scope, objectives, approach, resources, environments, schedule, roles, entry/exit criteria, risks, and deliverables.

20. What is a test strategy?

Answer: The higher-level approach defining how testing will be performed, including levels, types, environments, automation, and risk approach.

21. What are entry criteria?

Answer: Conditions that should be satisfied before a testing activity begins, such as an available build, environment, data, and required documentation.

22. What are exit criteria?

Answer: Conditions used to decide whether testing can stop, such as acceptable residual risk, required coverage, and critical tests passing.

23. What is a test closure report?

Answer: A summary of testing performed, coverage, defect status, deviations, metrics, unresolved risks, and release conclusions.

24. When should testing start?

Answer: As early as practical. Testers can review requirements and designs before executable software exists.

25. What is shift-left testing?

Answer: Moving quality activities earlier, such as requirement reviews, acceptance-criteria design, static checks, and early component/API testing.

26. What is shift-right testing?

Answer: Using production feedback, monitoring, observability, experiments, and real-user behavior to continue learning about quality after release.

27. What is Agile testing?

Answer: Continuous testing and collaboration throughout short iterations rather than treating testing as a separate phase at the end.

28. Agile vs Waterfall testing?

Answer: Waterfall generally separates phases, while Agile integrates development and testing continuously with frequent feedback and regression.

Testing Types and Levels

29. What is functional testing?

Answer: Testing whether software behavior matches specified functional requirements.

30. What is non-functional testing?

Answer: Testing quality attributes such as performance, usability, reliability, security, compatibility, and accessibility.

31. What is unit testing?

Answer: Testing a small software unit, commonly in isolation, usually performed by developers.

32. What is integration testing?

Answer: Testing interactions and interfaces between components, services, modules, or systems.

33. What is system testing?

Answer: Testing the integrated application as a complete system against specified requirements.

34. What is acceptance testing?

Answer: Testing whether the system meets business/user acceptance criteria and is suitable for delivery.

35. What is smoke testing?

Answer: A broad, shallow check that critical functionality works and the build is stable enough for deeper testing.

36. What is sanity testing?

Answer: A focused check of a changed or fixed area to determine whether it behaves as expected.

37. Smoke vs sanity testing?

Answer: Smoke testing is broad and checks build stability; sanity testing is narrow and focuses on a specific change or area.

38. What is regression testing?

Answer: Re-executing selected tests to detect unintended effects of changes on previously working functionality.

39. What is retesting?

Answer: Running a failed test again after a fix to determine whether the specific defect has been resolved.

40. Retesting vs regression?

Answer: Retesting validates a specific fix; regression checks whether changes caused unintended failures elsewhere.

41. What is exploratory testing?

Answer: A simultaneous learning, test-design, and execution approach where the tester adapts tests based on observations.

42. What is ad hoc testing?

Answer: Informal testing without detailed predefined test cases, often used for quick probing. It is less structured than exploratory testing.

43. What is compatibility testing?

Answer: Testing across supported browsers, operating systems, devices, databases, networks, or other relevant platforms.

44. What is usability testing?

Answer: Evaluating how effectively, efficiently, and satisfactorily intended users can accomplish tasks.

45. What is accessibility testing?

Answer: Checking whether people with disabilities can use the product, including keyboard access, focus, semantics, and assistive-technology behavior.

46. What is recovery testing?

Answer: Testing whether the system can recover correctly after failures such as crashes, network interruptions, or service loss.

Test Design Techniques

47. What is equivalence partitioning?

Answer: Dividing inputs into classes expected to behave similarly and testing representative values from each class.

48. What is boundary value analysis?

Answer: Testing values at and around boundaries because defects commonly occur at limits.

49. Give a boundary-value example.

Answer: For an age field accepting 18–60, useful values include 17, 18, 19, 59, 60, and 61.

50. What is decision table testing?

Answer: A matrix of conditions and resulting actions used to test combinations of business rules.

51. What is state transition testing?

Answer: Testing how a system behaves when it moves between states in response to events or conditions.

52. What is use-case testing?

Answer: Deriving tests from user interactions, including normal, alternate, and exception flows.

53. What is pairwise testing?

Answer: A combinatorial technique designed to cover all relevant pairs of parameter values while reducing combinations.

54. What is error guessing?

Answer: Using tester experience and knowledge of common failure patterns to target likely defect areas.

55. What is risk-based testing?

Answer: Prioritizing testing according to the probability and impact of failure.

56. How would you test a field accepting 1–100?

Answer: Test valid boundaries and representatives such as 1, 50, and 100; invalid values such as 0 and 101; and relevant blanks, spaces, decimals, very large values, and non-numeric input.

57. How do you test an ambiguous requirement?

Answer: Identify the ambiguity, document assumptions, ask stakeholders for clarification, and avoid silently inventing expected behavior.

58. How do you ensure coverage?

Answer: Map tests to requirements, prioritize risks, include positive/negative/boundary cases, and review untested conditions and paths.

Positive, Negative and Edge Cases

59. What is positive testing?

Answer: Testing valid inputs and expected user behavior to confirm intended functionality.

60. What is negative testing?

Answer: Testing invalid, unexpected, malformed, unauthorized, or unacceptable inputs and conditions.

61. What is edge-case testing?

Answer: Testing unusual or extreme conditions near operational limits.

62. Why is negative testing important?

Answer: It exposes validation gaps, error-handling defects, security weaknesses, and hidden assumptions.

63. How would you test a login page?

Answer: Test valid/invalid credentials, empty fields, boundaries, whitespace, locked users, case sensitivity, password masking, sessions, errors, accessibility, compatibility, and security controls.

64. How would you test a password field?

Answer: Check valid/invalid lengths, allowed characters, masking, whitespace, Unicode, expiry/reset rules, confirmation mismatch, strength rules, and secure handling.

65. How would you test a date field?

Answer: Check valid/invalid dates, leap years, boundaries, formats, locale, timezone effects, allowed ranges, empty values, and picker/manual entry.

66. How would you test a search box?

Answer: Test exact/partial search, case, whitespace, special characters, no results, long input, empty input, pagination, sorting/filter interaction, and performance.

Defects and Tricky Questions

67. What belongs in a defect report?

Answer: Clear title, environment/build, preconditions, steps, test data, expected result, actual result, evidence, severity, reproducibility, and logs/IDs where relevant.

68. Severity vs priority?

Answer: Severity describes impact; priority describes urgency or business order in which the issue should be addressed.

69. Can low severity have high priority?

Answer: Yes. A cosmetic issue on a highly visible launch page can be technically minor but commercially urgent.

70. Can high severity have low priority?

Answer: Yes, if it affects a rarely used area or is outside the current release scope, provided the risk is explicitly accepted.

71. What is a blocker defect?

Answer: A problem that prevents meaningful testing or progress, such as a critical environment or application failure.

72. What is a duplicate defect?

Answer: A newly reported issue representing the same underlying problem as an existing defect.

73. What does cannot reproduce mean?

Answer: The behavior cannot currently be reproduced under available conditions. Investigate environment, data, timing, concurrency, and configuration differences before dismissing it.

74. Developer says 'It works on my machine.' What do you do?

Answer: Compare versions, configuration, environment, data, dependencies, browser/device, feature flags, and steps. Provide a minimal reproducible case and evidence.

75. Developer rejects your defect. What do you do?

Answer: Discuss expected behavior using requirements, acceptance criteria, design decisions, or stakeholder clarification. Keep the discussion evidence-based.

76. You have one hour and 100 test cases. What do you do?

Answer: Do not execute randomly. Prioritize critical business flows, recent changes, integrations, high-risk areas, and smoke coverage, then communicate reduced coverage and residual risk.

77. A fix arrives minutes before release. What do you do?

Answer: Retest the fix and perform targeted regression around affected dependencies. If full regression is impossible, communicate the residual risk.

78. A test passes but the application is clearly wrong. What do you do?

Answer: Recheck the requirement, expected result, data, environment, and test implementation. A passing test does not prove product correctness.

79. A test fails but you suspect the test is wrong. What do you do?

Answer: Investigate the requirement, test data, environment, dependencies, and test logic before raising a product defect.

80. Requirements are incomplete. Can you test?

Answer: Yes, for well-defined behavior and exploratory/risk analysis, but formal pass/fail decisions should not rely on unresolved acceptance criteria.

81. Should a tester ever hide a defect to meet a release date?

Answer: No. Report objective evidence and risk. Release decisions can accept risk; test results should not be falsified.

Web and UI Manual Testing

82. What should you test on a web application?

Answer: Functionality, navigation, forms, validation, sessions, permissions, links, browser compatibility, responsiveness, accessibility, errors, and relevant security/performance behavior.

83. How do you test a broken link?

Answer: Verify the target, response/redirect behavior, authentication requirements, expected destination, and behavior across supported browsers.

84. How do you test a dropdown?

Answer: Check default value, options, ordering, selection, keyboard use, long lists, disabled options, multi-select behavior, persistence, and validation.

85. How do you test pagination?

Answer: Check first/last pages, next/previous, page numbers, page-size changes, empty results, boundaries, item counts, sorting/filtering, URL state, and data changes.

86. How do you test file upload?

Answer: Test supported/unsupported types, size boundaries, empty/corrupt files, duplicate names, multiple files, special names, cancellation, permissions, validation, and post-upload behavior.

87. How do you test responsive design?

Answer: Check supported viewport sizes/devices for layout, navigation, text wrapping, touch targets, overflow, forms, orientation, and content visibility.

88. What is cross-browser testing?

Answer: Verifying supported functionality and presentation across the browsers and versions defined by the product compatibility matrix.

89. How do cookies and sessions affect testing?

Answer: They affect authentication, persistence, personalization, expiration, logout, multi-tab behavior, and security. Test creation, expiry, invalidation, and state handling.

90. How would you test a broken image?

Answer: Check source URL, fallback behavior, dimensions, alt text where applicable, caching/CDN behavior, responsive rendering, and behavior when the resource is unavailable.

91. How would you test a form?

Answer: Validate mandatory fields, formats, boundaries, dependencies, error messages, keyboard navigation, submission behavior, duplicate clicks, reset behavior, and server-side validation.

API and Backend Manual Testing

92. Can a manual tester test APIs?

Answer: Yes. Tools such as Postman or command-line clients can validate requests, responses, status codes, headers, authentication, schema, business rules, and errors.

93. What should you validate in an API response?

Answer: Status, headers, body, schema, types, required fields, business rules, error structure, timing where relevant, and sensitive-data exposure.

94. 200 OK but wrong data: pass or fail?

Answer: Fail. A successful HTTP status does not mean the business response is correct.

95. 200 OK with an empty body: defect?

Answer: It depends on the contract. If a body is expected, it is a defect; if an empty body is documented, it can be correct.

96. 400 vs 404?

Answer: 400 generally indicates an invalid/malformed request; 404 generally indicates the requested resource or endpoint cannot be found, subject to the API contract.

97. 401 vs 403?

Answer: 401 generally concerns missing/invalid authentication; 403 generally means access is forbidden for the authenticated principal.

98. PUT vs PATCH?

Answer: PUT commonly represents replacement and is generally intended to be idempotent; PATCH is intended for partial modification. Exact semantics depend on the API contract.

99. What is idempotency?

Answer: Repeating an idempotent operation produces the same intended final state as performing it once, even if responses differ.

100. How do you test an API when the UI is unavailable?

Answer: Use the API contract or observed behavior and validate positive, negative, boundary, authorization, validation, dependency, and data-consistency scenarios.

101. How do you test an API without documentation?

Answer: Use authorized traffic, existing clients/tests, developer knowledge, and observed behavior to infer the contract, while documenting assumptions.

102. How do you test API authentication?

Answer: Test missing credentials, invalid credentials, expired tokens, wrong scopes/roles, token reuse, logout/revocation, and successful authorized access.

103. How do you test rate limiting?

Answer: Send controlled repeated requests and verify the documented threshold, response behavior, retry guidance, reset behavior, and that legitimate traffic recovers.

104. How do you test an API error response?

Answer: Validate status code, error schema, message/code, required fields, absence of sensitive data, consistency, and client-relevant remediation information.

Database and Data Validation

105. Why should a tester understand SQL?

Answer: SQL helps validate persistence, investigate defects, prepare controlled data, verify transactions, and compare UI/API behavior with stored data.

106. UI shows success but database record is missing. What do you check?

Answer: Verify environment, transaction completion, asynchronous processing, filters, replication delay, permissions, test data, and what the UI's success state actually means.

107. What is data integrity?

Answer: Data integrity means data remains accurate, consistent, complete, and trustworthy.

108. What is a primary key?

Answer: A column or column set that uniquely identifies a row and does not allow duplicate key values.

109. What is a foreign key?

Answer: A column or column set that references a key in another table to represent a relationship.

110. What is a transaction?

Answer: A logical unit of database work that should satisfy properties such as atomicity, consistency, isolation, and durability where applicable.

111. How do you test duplicate records?

Answer: Identify the uniqueness rule, submit equivalent data repeatedly, and verify whether duplicates are rejected, merged, or intentionally allowed.

112. How do you validate data end to end?

Answer: Trace controlled data from source through UI/API, processing, database, downstream systems, and final presentation, checking identifiers and transformations.

113. How would you test data migration?

Answer: Compare source and target counts and representative records, validate mappings, transformations, constraints, nulls, duplicates, rejected records, and reconciliation totals.

Agile and Scrum

114. What is a sprint?

Answer: A fixed period in which the team plans, builds, tests, reviews, and potentially delivers a product increment.

115. What is Definition of Done?

Answer: A shared agreement describing what must be true for work to be considered complete.

116. What is acceptance criteria?

Answer: Specific conditions that a story or feature must satisfy to be accepted.

117. What happens in backlog refinement?

Answer: The team clarifies requirements, identifies edge cases/dependencies, improves acceptance criteria, estimates work, and discusses risks and testability.

118. What should a tester contribute during refinement?

Answer: Ask clarifying questions, identify edge cases and dependencies, challenge ambiguity, define testability concerns, and help shape acceptance criteria.

119. What is a user story?

Answer: A concise description of desired functionality from a user's or stakeholder's perspective, normally accompanied by acceptance criteria.

120. A story is marked done but has known defects. What do you do?

Answer: Check whether they violate the Definition of Done or acceptance criteria. Track unresolved defects explicitly and discuss release risk.

121. What metrics are useful?

Answer: Execution results, defect trends, severity distribution, escaped defects, coverage, cycle time, and regression outcomes—used to support decisions rather than vanity targets.

Scenario-Based and Senior Questions

122. How would you test a calculator?

Answer: Test operations, zero, negatives, decimals, large values, precision, division by zero, chained operations, precedence, clear/history, keyboard input, localization, and errors.

123. How would you test an ATM?

Answer: Test authentication, balance, withdrawal/deposit, limits, invalid PIN, blocked card, insufficient funds, cash availability, network failure, receipt, timeout, session termination, and transaction consistency.

124. How would you test e-commerce checkout?

Answer: Cover cart accuracy, pricing, discounts, taxes, shipping, inventory, payment success/failure, retries, duplicate submission, order creation, confirmation, cancellation/refunds, and security.

125. How would you test a traffic-light controller?

Answer: Model states and transitions, timing, sensors, conflicting signals, pedestrian mode, emergency overrides, power/network failure, recovery, and safe-state behavior.

126. How would you test OTP login?

Answer: Test valid/invalid/expired OTPs, resend limits, multiple requests, old OTP invalidation, wrong number, rate limiting, lockout, restart, network interruption, accessibility, and secure handling.

127. What if a requirement only says 'user can search'?

Answer: Clarify searchable fields, exact/partial matching, case sensitivity, pagination, sorting, no results, special characters, performance, permissions, and error behavior.

128. How do you prioritize test cases?

Answer: Use business criticality, customer impact, failure probability, change impact, integration risk, historical defects, and security/regulatory risk.

129. How do you decide whether to automate a test?

Answer: Consider repeatability, stability, frequency, setup/data cost, maintenance, feedback value, determinism, and ROI.

130. What should generally not be automated first?

Answer: Highly unstable features, one-time exploratory work, rapidly changing UI, subjective usability checks, and low-value tests with high maintenance cost.

131. Production has a critical defect. What do you do?

Answer: Help reproduce and characterize impact, support triage, identify affected data/users, validate the fix, perform targeted regression, and contribute to root-cause analysis.

132. A manager asks you to mark failed tests as passed. What do you do?

Answer: Do not falsify results. Report objective evidence, explain residual risk, and let authorized stakeholders make an explicit release decision.

133. How do you know when to stop testing?

Answer: When exit criteria are met and remaining risk is understood and accepted. Passing all scripted tests alone does not prove zero risk.

134. How do you build a test strategy for a new product?

Answer: Understand business goals and architecture, identify risks, define levels/types, environments/data, coverage, automation boundaries, entry/exit criteria, defect workflow, and reporting.

135. How do you test asynchronous systems?

Answer: Test state transitions, eventual consistency, delays, duplicate messages, ordering, retries, dead-letter behavior, partial failure, idempotency, and recovery.

136. How do you test third-party integrations?

Answer: Validate contracts, authentication, positive/negative responses, timeouts, retries, rate limits, mappings, version compatibility, dependency failure, and observability.

137. What is defect leakage?

Answer: A defect that escapes one testing stage and is discovered later, for example in production.

138. What is root-cause analysis?

Answer: A structured investigation into why a defect occurred, considering code, requirements, process, test gaps, data, environment, and integration causes.

139. Is 100% test coverage possible?

Answer: Not as a guarantee of every possible behavior and combination. Aim for sufficient risk-based coverage for the product context.

140. Is zero defects realistic?

Answer: Zero known defects may be a release goal, but it does not prove zero undiscovered defects. The practical objective is acceptable residual risk.

141. What is the biggest mistake a manual tester can make?

Answer: Treating execution of scripted test cases as the entire job. Strong testers question requirements, explore risk, investigate failures, and communicate product risk.

Final Interview Checklist

- Explain verification vs validation with a practical example.
- Differentiate smoke, sanity, regression and retesting.
- Design positive, negative, boundary and edge-case tests.
- Explain severity vs priority with examples.
- Write a complete defect report from a vague problem statement.
- Test login, search, forms, upload, pagination and checkout scenarios.
- Validate an API beyond checking its HTTP status code.
- Explain how UI/API data can be validated against a database.
- Prioritize testing when time and resources are limited.
- Handle disagreement with a developer using evidence and requirements.
- Describe a difficult defect you found, isolated and communicated.
- Explain your release-risk decision when testing cannot be exhaustive.

Interview principle: The strongest answers demonstrate reasoning rather than memorized definitions. When a question is ambiguous, clarify the requirement, state assumptions, identify risk, and then propose tests.