Manual Testing Interview Questions With Answer

Introduction

Manual testing remains the backbone of quality assurance, especially in agile and exploratory contexts. Whether you are a fresh graduate or an experienced QA professional, preparing for an interview requires a solid grasp of core concepts, practical scenarios, and the ability to articulate your thought process. This article provides a comprehensive set of **Manual Testing Interview Questions With Answer** that cover foundational knowledge, test design techniques, defect management, and real-world problem-solving. Each answer is crafted to be clear, insightful, and ready to use in your next interview. Let’s dive in.

Part 1 – Core Concepts of Manual Testing

Interviewers often begin with basic definitions to gauge your understanding of the discipline. Below are essential questions and answers that every manual tester should know.

Q1: What is manual testing? How does it differ from automated testing?

Answer: Manual testing involves executing test cases without the help of automation tools. The tester plays the role of an end user and validates the application’s behavior, UI, and functionality. In contrast, automated testing uses scripts and tools (e.g., Selenium, Cypress) to run repetitive tests, which improves speed and accuracy for regression suites. Manual testing is essential for exploratory, usability, and ad-hoc testing where human judgment is irreplaceable.

Q2: Explain the Software Testing Life Cycle (STLC).

Answer: The STLC consists of six phases: **Requirement Analysis** (reviewing and prioritizing testable requirements), **Test Planning** (defining scope, resources, and schedule), **Test Case Development** (writing and reviewing test cases), **Test Environment Setup** (configuring hardware, software, and data), **Test Execution** (running tests and logging defects), and **Test Cycle Closure** (analyzing metrics, preparing reports, and archiving artifacts). Each phase has entry and exit criteria to ensure quality.

Q3: What is the difference between verification and validation?

Answer: Verification is the process of evaluating work products (requirements, design, test cases) to ensure they meet specified conditions – often summarized as “are we building the product right?” Validation, on the other hand, involves testing the final product against user needs – “are we building the right product?” Both are complementary; verification catches early defects, while validation ensures customer satisfaction.

Part 2 – Test Case Design Techniques

Interviewers frequently ask about design techniques to assess your ability to create effective test cases with optimal coverage. Here are typical **Manual Testing Interview Questions With Answer** in this area.

Q4: What is equivalence partitioning? Give an example.

Answer: Equivalence partitioning is a black-box technique that divides input data into partitions (or classes) from which test cases can be derived. Within each partition, values are expected to behave the same way. For example, a text field that accepts ages from 18 to 60 can be partitioned into three classes: invalid low (below 18), valid (18–60), and invalid high (above 60). One representative value from each class is sufficient for testing. This technique reduces redundancy and improves efficiency.

Q5: Explain boundary value analysis.

Answer: Boundary value analysis (BVA) focuses on values at the edges of equivalence partitions because defects often occur at boundaries. For the same age field (18–60), you would test at the boundaries: 17, 18, 19 (lower boundary) and 59, 60, 61 (upper boundary). BVA complements equivalence partitioning and is especially important for numeric ranges, date inputs, and string length limits.

Q6: What is a decision table? When would you use it?

Answer: A decision table is a matrix that lists all possible combinations of conditions (inputs) and the corresponding actions (outputs). It is used when the system’s behavior depends on multiple logical conditions, such as a login feature: conditions could be “valid username” and “valid password,” with actions like “grant access” or “show error.” Decision tables ensure complete coverage of business rules and help uncover missing logic.

Part 3 – Test Management and Documentation

Good documentation is the hallmark of a professional tester. Below are questions that test your knowledge of test planning, traceability, and reporting.

Q7: What is a test plan? List its key components.

Answer: A test plan is a comprehensive document that defines the scope, approach, resources, and schedule of testing activities. Key components include: test objectives, scope (in-scope and out-scope), test strategy (levels, types, techniques), resource allocation (team members, tools), environment requirements, test deliverables (cases, scripts, reports), risk analysis, and entry/exit criteria. A well-written test plan serves as a roadmap for the entire testing phase.

Q8: What is a traceability matrix (RTM)? How is it useful?

Answer: A requirement traceability matrix (RTM) maps test cases back to functional requirements. It ensures that every requirement has at least one corresponding test case and vice versa. The RTM is useful for verifying coverage during test execution, impact analysis when requirements change, and providing evidence to auditors or stakeholders that all requirements are tested.

Q9: What should a good test case include?

Answer: A well-structured test case contains: a unique ID, title, description, pre-conditions, test steps, test data, expected result, actual result (filled after execution), status (pass/fail), and severity/priority if it fails. Additional fields such as “test environment,” “dependencies,” and “regression flag” can improve traceability. Clear and reproducible steps are essential for test execution by different team members.

Part 4 – Defect Management

Understanding how to report and prioritize defects is critical. Here are common **Manual Testing Interview Questions With Answer** on defect lifecycle.

Q10: Explain the defect lifecycle (bug lifecycle).

Answer: The defect lifecycle typically includes the following states: **New** (bug is logged), **Assigned** (assigned to a developer), **Open** (developer starts working), **Fixed** (code change made), **Retest** (tester verifies fix), **Reopened** (if fix fails), **Verified** (fix confirmed), and **Closed**. Additional states like “Deferred” (postponed to a future release) or “Duplicate” may also exist in some tools. Every defect moves through this lifecycle, and clear communication is vital at each step.

Q11: What is the difference between severity and priority? Give an example.

Answer: Severity measures the impact on the application – how critical the bug is from a technical perspective (e.g., system crash vs. cosmetic typo). Priority measures the urgency of fixing the bug from a business or project perspective (e.g., a bug that blocks a major release vs. a minor UI glitch). For example, a misspelling on the company’s logo might have low severity (cosmetic) but high priority (branding). Conversely, a backend calculation error that only affects a rarely used feature might have high severity (data integrity) but low priority.

Q12: How do you write an effective bug report?

Answer: An effective bug report should be clear, concise, and reproducible. Include: a descriptive title, environment details (OS, browser, version), steps to reproduce (numbered and specific), actual result, expected result, screenshots or logs (if applicable), severity and priority, and any additional notes. Avoid vague statements like “the page is broken.” Instead, provide exact data and error messages to help developers quickly identify the root cause.

Part 5 – Scenario-Based Questions

Scenario-based questions assess your practical problem-solving skills. Below are typical scenarios that appear in manual testing interviews.

Q13: How would you test a login page?

Answer: Testing a login page requires covering both functional and non-functional aspects. First, test valid login with correct username and password. Then test invalid combinations: incorrect password, invalid username, blank fields, and both blank. Check boundary conditions for password length and character restrictions. Verify error messages, password masking, and “remember me” functionality. Also test session timeout, multiple failed attempts (account lockout), and cross-browser compatibility. For security, test SQL injection attempts in username and password fields. Finally, ensure the login page handles CAPTCHA or two-factor authentication if implemented.

Q14: What if there are no written requirements? How do you proceed?

Answer: In the absence of formal requirements, manual testers rely on **exploratory testing** and domain knowledge. Start by reviewing existing documentation (user manuals, mockups, similar applications). Conduct meetings with stakeholders (product owners, developers, business analysts) to clarify expected behavior. Use **heuristic evaluation** to identify common issues. Create test charters for each session that describe areas to explore. Document findings in real-time and prioritize testing based on risk. Also, leverage competitor products or industry standards as reference for expected functionality.

Q15: How do you decide which test cases to automate and which to keep manual?

Answer: The decision is based on several factors: test frequency (repetitive tests like regression are good for automation), test

complexity (complex UI interactions or visual verifications often remain manual), risk (critical paths may be automated for consistency), and maintenance cost (frequently changing features are better tested manually). Additionally, exploratory tests, usability tests, and one-time tests are inherently manual. A balanced approach uses automation for stable, high-volume tests and manual testing for creative, investigative scenarios.

Part 6 – Behavioral and Advanced Questions

Finally, interviewers look for soft skills and the ability to work in a team. Here are some behavioral **Manual Testing Interview Questions With Answer** that showcase your professionalism.

Q16: How do you handle disagreements with a developer about a bug?

Answer: I believe in respectful, data-driven communication. First, I double-check my reproduction steps and gather additional evidence (logs, screenshots, environment details). I then schedule a quick sync with the developer, present the facts, and explain the impact from the user's perspective. I listen to their technical constraints and explore potential trade-offs. If we still disagree, I escalate to a test lead or project manager with a clear summary of both arguments. The goal is to reach a decision that benefits product quality without creating friction.

Q17: Describe a time when you found a critical defect late in the release cycle. What did you do?

Answer: In one project, I discovered a data corruption bug during final regression that could affect millions of existing records. Immediately, I raised a high-severity, high-priority bug, communicated the impact to the release manager and QA lead, and suggested a rollback of the specific feature. I also helped the developer isolate the root cause by providing exact steps and database logs. The team decided to postpone the release by one day to fix the issue. My proactive communication prevented the defect from reaching production and saved the company from a major incident.

Q18: What are your strengths as a manual tester?

Answer: My strongest assets are **attention to detail** and **curiosity**. I take time to understand user workflows and business logic, which helps me identify edge cases that automation might miss.