

Testing Interview questions and answers

Manual Testing FAQs Part-I

Q: How do you introduce a new software QA process?

A: It depends on the size of the organization and the risks involved. For large organizations with high-risk projects, a serious management buy-in is required and a formalized QA process is necessary. For medium size organizations with lower risk projects, management and organizational buy-in and a slower, step-by-step process is required. Generally speaking, QA processes should be balanced with productivity, in order to keep any bureaucracy from getting out of hand. For smaller groups or projects, an ad-hoc process is more appropriate. A lot depends on team leads and managers, feedback to developers and good communication is essential among customers, managers, developers, test engineers and testers. Regardless the size of the company, the greatest value for effort is in managing requirement processes, where the goal is requirements that are clear, complete and testable.

Q: What is the role of documentation in QA?

A: Documentation plays a critical role in QA. QA practices should be documented, so that they are repeatable. Specifications, designs, business rules, inspection reports, configurations, code changes, test plans, test cases, bug reports, user manuals should all be documented. Ideally, there should be a system for easily finding and obtaining of documents and determining what document will have a particular piece of information. Use documentation change management, if possible.

Q: What makes a good test engineer?

A: Good test engineers have a "test to break" attitude. We, good test engineers, take the point of view of the customer; have a strong desire for quality and an attention to detail. Tact and diplomacy are useful in maintaining a cooperative relationship with developers and an ability to communicate with both technical and non-technical people. Previous software development experience is also helpful as it provides a deeper understanding of the software development process, gives the test engineer an appreciation for the developers' point of view and reduces the learning curve in automated test tool programming.

Rob Davis is a good test engineer because he has a "test to break" attitude, takes the point of view of the customer, has a strong desire for quality, has an attention to detail, He's also tactful and diplomatic and has good a communication skill, both oral and written. And he has previous software development experience, too.

Q: What is a test plan?

A: A software project test plan is a document that describes the objectives, scope, approach and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group understand the why and how of product validation. It should be thorough enough to be useful, but not so thorough that none outside the test group will be able to read it.

Testing Interview questions and answers

Q: What is a test case?

A: A test case is a document that describes an input, action, or event and its expected result, in order to determine if a feature of an application is working correctly. A test case should contain particulars such as a...

- Test case identifier;
- Test case name;
- Objective;
- Test conditions/setup;
- Input data requirements/steps, and
- Expected results.

Please note, the process of developing test cases can help find problems in the requirements or design of an application, since it requires you to completely think through the operation of the application. For this reason, it is useful to prepare test cases early in the development cycle, if possible.

Q: What should be done after a bug is found?

A: When a bug is found, it needs to be communicated and assigned to developers that can fix it. After the problem is resolved, fixes should be re-tested. Additionally, determinations should be made regarding requirements, software, hardware, safety impact, etc., for regression testing to check the fixes didn't create other problems elsewhere. If a problem-tracking system is in place, it should encapsulate these determinations. A variety of commercial, problem-tracking/management software tools are available. These tools, with the detailed input of software test engineers, will give the team complete information so developers can understand the bug, get an idea of its severity, reproduce it and fix it.

Q: What is configuration management?

A: Configuration management (CM) covers the tools and processes used to control, coordinate and track code, requirements, documentation, problems, change requests, designs, tools, compilers, libraries, patches, changes made to them and who makes the changes. Rob Davis has had experience with a full range of CM tools and concepts, and can easily adapt to your software tool and process needs.

Q: What if the software is so buggy it can't be tested at all?

A: In this situation the best bet is to have test engineers go through the process of reporting whatever bugs or problems initially show up, with the focus being on critical bugs.

Since this type of problem can severely affect schedules and indicates deeper problems in the software development process, such as insufficient unit testing, insufficient integration testing, poor design, improper build or release procedures, managers should be notified and provided with some documentation as evidence of the problem.

Testing Interview questions and answers

Q: What if there isn't enough time for thorough testing?

A: Since it's rarely possible to test every possible aspect of an application, every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to most software development projects.

Use risk analysis to determine where testing should be focused. This requires judgment skills, common sense and experience. The checklist should include answers to the following questions:

- Which functionality is most important to the project's intended purpose?
- Which functionality is most visible to the user?
- Which functionality has the largest safety impact?
- Which functionality has the largest financial impact on users?
- Which aspects of the application are most important to the customer?
- Which aspects of the application can be tested early in the development cycle?
- Which parts of the code are most complex and thus most subject to errors?
- Which parts of the application were developed in rush or panic mode?
- Which aspects of similar/related previous projects caused problems?
- Which aspects of similar/related previous projects had large maintenance expenses?
- Which parts of the requirements and design are unclear or poorly thought out?
- What do the developers think are the highest-risk aspects of the application?
- What kinds of problems would cause the worst publicity?
- What kinds of problems would cause the most customer service complaints?
- What kinds of tests could easily cover multiple functionalities?
- Which tests will have the best high-risk-coverage to time-required ratio?

Q: What if the project isn't big enough to justify extensive testing?

A: Consider the impact of project errors, not the size of the project. However, if extensive testing is still not justified, risk analysis is again needed and the considerations listed under "What if there isn't enough time for thorough testing?" do apply. The test engineer then should do "ad hoc" testing, or write up a limited test plan based on the risk analysis.

Q: What can be done if requirements are changing continuously?

A: Work with management early on to understand how requirements might change, so that alternate test plans and strategies can be worked out in advance. It is helpful if the application's initial design allows for some adaptability, so that later changes do not require redoing the application from scratch. Additionally, try to...

- Ensure the code is well commented and well documented; this makes changes easier for the developers.
- Use rapid prototyping whenever possible; this will help customers feel sure of their requirements and minimize changes.
- In the project's initial schedule, allow for some extra time to commensurate with probable changes.

Testing Interview questions and answers

Move new requirements to a 'Phase 2' version of an application and use the original requirements for the 'Phase 1' version.

Negotiate to allow only easily implemented new requirements into the project.

- Ensure customers and management understands scheduling impacts, inherent risks and costs of significant requirements changes. Then let management or the customers decide if the changes are warranted; after all, that's their job.
- Balance the effort put into setting up automated testing with the expected effort required to redo them to deal with changes.
- Design some flexibility into automated test scripts;
- Focus initial automated testing on application aspects that are most likely to remain unchanged;
- Devote appropriate effort to risk analysis of changes, in order to minimize regression-testing needs;
- Design some flexibility into test cases; this is not easily done; the best bet is to minimize the detail in the test cases, or set up only higher-level generic-type test plans;

Focus less on detailed test plans and test cases and more on ad-hoc testing with an understanding of the added risk this entails.

Q: How do you know when to stop testing?

A: This can be difficult to determine. Many modern software applications are so complex and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are...

- Deadlines, e.g. release deadlines, testing deadlines;
- Test cases completed with certain percentage passed;
- Test budget has been depleted;
- Coverage of code, functionality, or requirements reaches a specified point;
- Bug rate falls below a certain level; or
- Beta or alpha testing period ends.

Q: What if the application has functionality that wasn't in the requirements?

A: It may take serious effort to determine if an application has significant unexpected or hidden functionality, which it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed, as it may have unknown impacts or dependencies that were not taken into account by the designer or the customer.

If not removed, design information will be needed to determine added testing needs or regression testing needs. Management should be made aware of any significant added risks as a result of the unexpected functionality. If the functionality only affects areas, such as minor improvements in the user interface, it may not be a significant risk.

Testing Interview questions and answers

Q: How can software QA processes be implemented without stifling productivity?

A: Implement QA processes slowly over time. Use consensus to reach agreement on processes and adjust and experiment as an organization grows and matures. Productivity will be improved instead of stifled. Problem prevention will lessen the need for problem detection. Panics and burnout will decrease and there will be improved focus and less wasted effort.

At the same time, attempts should be made to keep processes simple and efficient, minimize paperwork, promote computer-based processes and automated tracking and reporting, minimize time required in meetings and promote training as part of the QA process.

However, no one, especially talented technical types, like bureaucracy and in the short run things may slow down a bit. A typical scenario would be that more days of planning and development will be needed, but less time will be required for late-night bug fixing and calming of irate customers.

Q: What if the organization is growing so fast that fixed QA processes are impossible?

A: This is a common problem in the software industry, especially in new technology areas. There is no easy solution in this situation, other than...

- Hire good people (i.e. hire Rob Davis)
- Ruthlessly prioritize quality issues and maintain focus on the customer;
- Everyone in the organization should be clear on what quality means to the customer.

Q: Why do you recommend that we test during the design phase?

A: Because testing during the design phase can prevent defects later on. We recommend verifying three things...

1. Verify the design is good, efficient, compact, testable and maintainable.
2. Verify the design meets the requirements and is complete (specifies all relationships between modules, how to pass data, what happens in exceptional circumstances, starting state of each module and how to guarantee the state of each module).
3. Verify the design incorporates enough memory, I/O devices and quick enough runtime for the final product.

Q: What is software quality assurance?

A: Software Quality Assurance, when Rob Davis does it, is oriented to *prevention*. It involves the entire software development process. Prevention is monitoring and improving the process, making sure any agreed-upon standards and procedures are followed and ensuring problems are found and dealt with.

Testing Interview questions and answers

Software Testing, when performed by Rob Davis, is also oriented to *detection*. Testing involves the operation of a system or application under controlled conditions and evaluating the results.

Rob Davis can provide QA/testing service. This document details some aspects of how he can provide software testing/QA service. For more information, e-mail rob@robdavispe.com.

Organizations vary considerably in how they assign responsibility for QA and testing. Sometimes they're the combined responsibility of one group or individual.

Also common are project teams, which include a mix of test engineers, testers and developers, who work closely together, with overall QA processes monitored by project managers.

Software quality assurance depends on what best fits your organization's size and business structure.

Q: How is testing affected by object-oriented designs?

A: A well-engineered object-oriented design can make it easier to trace from code to internal design to functional design to requirements. While there will be little affect on black box testing (where an understanding of the internal design of the application is unnecessary), white-box testing can be oriented to the application's objects. If the application was well designed this can simplify test design.

Q: What is quality assurance?

A: Quality Assurance ensures all parties concerned with the project adhere to the process and procedures, standards and templates and test readiness reviews.

Rob Davis' QA service depends on the customers and projects. A lot will depend on team leads or managers, feedback to developers and communications among customers, managers, developers' test engineers and testers.

Q: What is black box testing?

A: Black box testing is functional testing, not based on any knowledge of internal software design or code. Black box testing is based on requirements and functionality.

Q: What is white box testing?

A: White box testing is based on knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths and conditions.

Q: What is unit testing?

Testing interview questions and answers for Freshers and experiences! 6

Testing Interview questions and answers

A: Unit testing is the first level of dynamic testing and is first the responsibility of developers and then that of the test engineers.

Unit testing is performed after the expected test results are met or differences are explainable/ acceptable.

Q: What is functional testing?

A: Functional testing is black-box type of testing geared to functional requirements of an application. Test engineers **should** perform functional testing.

Q: What is usability testing?

A: Usability testing is testing for 'user-friendliness'. Clearly this is subjective and depends on the targeted end-user or customer. User interviews, surveys, video recording of user sessions and other techniques can be used. Programmers and developers are usually not appropriate as usability testers.

Q: What is incremental integration testing?

A: Incremental integration testing is continuous testing of an application as new functionality is recommended. This may require that various aspects of an application's functionality are independent enough to work separately, before all parts of the program are completed, or that test drivers are developed as needed.

Incremental testing may be performed by programmers, software engineers, or test engineers.

Q: What is parallel/audit testing?

A: Parallel/audit testing is testing where the user reconciles the output of the new system to the output of the current system to verify the new system performs the operations correctly.

Q: What is integration testing?

A: Upon completion of unit testing, integration testing begins. Integration testing is black box testing. The purpose of integration testing is to ensure distinct components of the application still work in accordance to customer requirements.

Test cases are developed with the express purpose of exercising the interfaces between the components. This activity is carried out by the test team.

Integration testing is considered complete, when actual results and expected results are either in line or differences are explainable/acceptable based on client input.

Testing Interview questions and answers

Q: What is system testing?

A: System testing is black box testing, performed by the Test Team, and at the start of the system testing the complete system is configured in a controlled environment.

The purpose of system testing is to validate an application's accuracy and completeness in performing the functions as designed.

System testing simulates real life scenarios that occur in a "simulated real life" test environment and test all functions of the system that are required in real life.

System testing is deemed complete when actual results and expected results are either in line or differences are explainable or acceptable, based on client input.

Upon completion of integration testing, system testing is started. Before system testing, all unit and integration test results are reviewed by Software QA to ensure all problems have been resolved. For a higher level of testing it is important to understand unresolved problems that originate at unit and integration test levels.

You CAN learn system testing, with little or no outside help. Get CAN get free information. Click on a link!

Q: What is end-to-end testing?

A: Similar to system testing, the *macro* end of the test scale is testing a complete application in a situation that mimics real world use, such as interacting with a database, using network communication, or interacting with other hardware, application, or system.

Q: What is regression testing?

A: The objective of regression testing is to ensure the software remains intact. A baseline set of data and scripts is maintained and executed to verify changes introduced during the release have not "undone" any previous code. Expected results from the baseline are compared to results of the software under test. All discrepancies are highlighted and accounted for, before testing proceeds to the next level.

Q: What is sanity testing?

A: Sanity testing is performed whenever cursory testing is sufficient to prove the application is functioning according to specifications. This level of testing is a subset of regression testing.

It normally includes a set of core tests of basic GUI functionality to demonstrate connectivity to the database, application servers, printers, etc.

Q: What is performance testing?

A: Although performance testing is described as a part of system testing, it can

Testing interview questions and answers for Freshers and 8 experiences!

Testing Interview questions and answers

be regarded as a distinct level of testing. Performance testing verifies loads, volumes and response times, as defined by requirements.

Q: What is load testing?

A: Load testing is testing an application under heavy loads, such as the testing of a web site under a range of loads to determine at what point the system response time will degrade or fail.

Q: What is installation testing?

A: Installation testing is testing full, partial, upgrade, or install/uninstall processes. The installation test for a release is conducted with the objective of demonstrating production readiness.

This test includes the inventory of configuration items, performed by the application's System Administration, the evaluation of data readiness, and dynamic tests focused on basic system functionality. When necessary, a sanity test is performed, following installation testing.

Q: What is security/penetration testing?

A: Security/penetration testing is testing how well the system is protected against unauthorized internal or external access, or willful damage.

This type of testing usually requires sophisticated testing techniques.

Q: What is recovery/error testing?

A: Recovery/error testing is testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Q: What is compatibility testing?

A: Compatibility testing is testing how well software performs in a particular hardware, software, operating system, or network

This test includes the inventory of configuration items, performed by the application's System Administration, the evaluation of data readiness, and dynamic tests focused on basic system functionality. When necessary, a sanity test is performed, following installation testing.

Q: What is comparison testing?

A: Comparison testing is testing that compares software weaknesses and strengths to those of competitors' products.

Q: What is acceptance testing?

Testing Interview questions and answers

A: Acceptance testing is black box testing that gives the client/customer/project manager the opportunity to verify the system functionality and usability prior to the system being released to production.

The acceptance test is the responsibility of the client/customer or project manager, however, it is conducted with the full support of the project team. The test team also works with the client/customer/project manager to develop the acceptance criteria.

Q: What is alpha testing?

A: Alpha testing is testing of an application when development is nearing completion. Minor design changes can still be made as a result of alpha testing. Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.

Q: What is beta testing?

A: Beta testing is testing an application when development and testing are essentially completed and final bugs and problems need to be found before the final release. Beta testing is typically performed by end-users or others, not programmers, software engineers, or test engineers.

Q: What is a Test/QA Team Lead?

A: The Test/QA Team Lead coordinates the testing activity, communicates testing status to management and manages the test team.

Q: What testing roles are standard on most testing projects?

A: Depending on the organization, the following roles are more or less standard on most testing projects: Testers, Test Engineers, Test/QA Team Lead, Test/QA Manager, System Administrator, Database Administrator, Technical Analyst, Test Build Manager and Test Configuration Manager.

Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of Technical Analyst, Test Build Manager and Test Configuration Manager.

You CAN get a job in testing. Click on a link!

Q: What is a Test Engineer?

A: We, test engineers, are engineers who specialize in testing. We, test engineers, create test cases, procedures, scripts and generate data. We execute test procedures and scripts, analyze standards of measurements, evaluate results of system/integration/regression testing. We also...

- Speed up the work of the development staff;
- Reduce your organization's risk of legal liability;
- Give you the evidence that your software is correct and operates properly;

Testing interview questions and answers for Freshers and experiences! 10

Testing Interview questions and answers

- Improve problem tracking and reporting;
- Maximize the value of your software;
- Maximize the value of the devices that use it;
- Assure the successful launch of your product by discovering bugs and design flaws, before users get discouraged, before shareholders lose their cool and before employees get bogged down;
- Help the work of your development staff, so the development team can devote its time to build up your product;
- Promote continual improvement;
- Provide documentation required by FDA, FAA, other regulatory agencies and your customers;
- Save money by discovering defects 'early' in the design process, before failures occur in production, or in the field;
- Save the reputation of your company by discovering bugs and design flaws; before bugs and design flaws damage the reputation of your company.

: What is a Test Build Manager?

A: Test Build Managers deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware.

Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a Test Build Manager.

Q: What is a System Administrator?

A: Test Build Managers, System Administrators, Database Administrators deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware.

Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a System Administrator.

Q: What is a Database Administrator?

A: Test Build Managers, System Administrators and Database Administrators deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware. Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a Database Administrator.

Q: What is a Technical Analyst?

A: Technical Analysts perform test assessments and validate system/functional test requirements. Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of a Technical Analyst.

Testing Interview questions and answers

Q: What is a Test Configuration Manager?

A: Test Configuration Managers maintain test environments, scripts, software and test data. Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of a Test Configuration Manager.

Q: What is a test schedule?

A: The test schedule is a schedule that identifies all tasks required for a successful testing effort, a schedule of all test activities and resource requirements.

Q: What is software testing methodology?

A: One software testing methodology is the use a three step process of...

1. Creating a test strategy;
2. Creating a test plan/design; and
3. Executing tests.

This methodology can be used and molded to your organization's needs. Rob Davis believes that using this methodology is important in the development and ongoing maintenance of his clients' applications.

Q: What is the general testing process?

A: The general testing process is the creation of a test strategy (which sometimes includes the creation of test cases), creation of a test plan/design (which usually includes test cases and test procedures) and the execution of tests.

Q: How do you create a test plan/design?

A: Test scenarios and/or cases are prepared by reviewing functional requirements of the release and preparing logical groups of functions that can be further broken into test procedures. Test procedures define test conditions, data to be used for testing and expected results, including database updates, file outputs, report results. Generally speaking...

- Test cases and scenarios are designed to represent both typical and unusual situations that may occur in the application.
- Test engineers define unit test requirements and unit test cases. Test engineers also execute unit test cases.
- It is the test team that, with assistance of developers and clients, develops test cases and scenarios for integration and system testing.
- Test scenarios are executed through the use of test procedures or scripts.
- Test procedures or scripts define a series of steps necessary to perform one or more test scenarios.
- Test procedures or scripts include the specific data that will be used for testing the process or transaction.

Testing Interview questions and answers

- Test procedures or scripts may cover multiple test scenarios.
- Test scripts are mapped back to the requirements and traceability matrices are used to ensure each test is within scope.
- Test data is captured and base lined, prior to testing. This data serves as the foundation for unit and system testing and used to exercise system functionality in a controlled environment.
- Some output data is also base-lined for future comparison. Base-lined data is used to support future application maintenance via regression testing.
- A pretest meeting is held to assess the readiness of the application and the environment and data to be tested. A test readiness document is created to indicate the status of the entrance criteria of the release.

Inputs for this process:

- Approved Test Strategy Document.
- Test tools, or automated test tools, if applicable.
- Previously developed scripts, if applicable.
- Test documentation problems uncovered as a result of testing.
- A good understanding of software complexity and module path coverage, derived from general and detailed design documents, e.g. software design document, source code, and software complexity data.

Outputs for this process:

- Approved documents of test scenarios, test cases, test conditions, and test data.
- Reports of software design issues, given to software developers for correction.

Q: How do you execute tests?

A: Execution of tests is completed by following the test documents in a methodical manner. As each test procedure is performed, an entry is recorded in a test execution log to note the execution of the procedure and whether or not the test procedure uncovered any defects. Checkpoint meetings are held throughout the execution phase. Checkpoint meetings are held daily, if required, to address and discuss testing issues, status and activities.

- The output from the execution of test procedures is known as test results. Test results are evaluated by test engineers to determine whether the expected results have been obtained. All discrepancies/anomalies are logged and discussed with the software team lead, hardware test lead, programmers, software engineers and documented for further investigation and resolution. Every company has a different process for logging and reporting bugs/defects uncovered during testing.
- A pass/fail criteria is used to determine the severity of a problem, and results are recorded in a test summary report. The severity of a problem, found during system testing, is defined in accordance to the customer's risk assessment and recorded in their selected tracking tool.
- Proposed fixes are delivered to the testing environment, based on the severity of the problem. Fixes are regression tested and flawless fixes are migrated to a new baseline. Following completion of the test, members of the test team prepare a summary report. The summary report is reviewed by the Project Manager, Software QA Manager and/or Test Team Lead.

Testing Interview questions and answers

- After a particular level of testing has been certified, it is the responsibility of the Configuration Manager to coordinate the migration of the release software components to the next test level, as documented in the Configuration Management Plan. The software is only migrated to the production environment after the Project Manager's formal acceptance.
- The test team reviews test document problems identified during testing, and update documents where appropriate.

Inputs for this process:

- Approved test documents, e.g. Test Plan, Test Cases, Test Procedures.
- Test tools, including automated test tools, if applicable.
- Developed scripts.
- Changes to the design, i.e. Change Request Documents.
- Test data.
- Availability of the test team and project team.
- General and Detailed Design Documents, i.e. Requirements Document, Software Design Document.
- A software that has been migrated to the test environment, i.e. unit tested code, via the Configuration/Build Manager.
- Test Readiness Document.
- Document Updates.

Outputs for this process:

- Log and summary of the test results. Usually this is part of the Test Report. This needs to be approved and signed-off with revised testing deliverables.
- Changes to the code, also known as test fixes.
- Test document problems uncovered as a result of testing. Examples are Requirements document and Design Document problems.
- Reports on software design issues, given to software developers for correction. Examples are bug reports on code issues.
- Formal record of test incidents, usually part of problem tracking.
- Base-lined package, also known as tested source and object code, ready for migration to the next level.

Q: How do you create a test strategy?

A: The test strategy is a formal description of how a software product will be tested. A test strategy is developed for all levels of testing, as required. The test team analyzes the requirements, writes the test strategy and reviews the plan with the project team. The test plan may include test cases, conditions, the test environment, a list of related tasks, pass/fail criteria and risk assessment.

Inputs for this process:

- A description of the required hardware and software components, including test tools. This information comes from the test environment, including test tool data.
- A description of roles and responsibilities of the resources required for the test and schedule constraints. This information comes from man-hours and schedules.
- Testing methodology. This is based on known standards.

Testing Interview questions and answers

- Functional and technical requirements of the application. This information comes from requirements, change request, technical and functional design documents.
- Requirements that the system can not provide, e.g. system limitations.

Outputs for this process:

- An approved and signed off test strategy document, test plan, including test cases.
- Testing issues requiring resolution. Usually this requires additional negotiation at the project management level.

Q: What is security clearance?

A: Security clearance is a process of determining your trustworthiness and reliability before granting you access to national security information.

Q: What are the levels of classified access?

A: The levels of classified access are confidential, secret, top secret, and sensitive compartmented information, of which top secret is the highest.

What's a 'test plan'?

A software project test plan is a document that describes the objectives, scope, approach, and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group understand the 'why' and 'how' of product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it. The following are some of the items that might be included in a test plan, depending on the particular project:

- * Title
- * Identification of software including version/release numbers.
- * Revision history of document including authors, dates, approvals.
- * Table of Contents.
- * Purpose of document, intended audience
- * Objective of testing effort
- * Software product overview
- * Relevant related document list, such as requirements, design documents, other test plans, etc.
- * Relevant standards or legal requirements

Testing Interview questions and answers

- * Traceability requirements
- * Relevant naming conventions and identifier conventions
- * Overall software project organization and personnel/contact-info/responsibilities
- * Test organization and personnel/contact-info/responsibilities
- * Assumptions and dependencies
- * Project risk analysis
- * Testing priorities and focus
- * Scope and limitations of testing
- * Test outline - a decomposition of the test approach by test type, feature, functionality, process, system, module, etc. as applicable
- * Outline of data input equivalence classes, boundary value analysis, error classes
- * Test environment - hardware, operating systems, other required software, data configurations, interfaces to other systems
- * Test environment validity analysis - differences between the test and production systems and their impact on test validity.
- * Test environment setup and configuration issues
- * Software migration processes
- * Software CM processes
- * Test data setup requirements
 - * Database setup requirements
 - * Outline of system-logging/error-logging/other capabilities, and tools such as screen capture software, that will be used to help describe and report bugs
 - * Discussion of any specialized software or hardware tools that will be used by testers to help track the cause or source of bugs
 - * Test automation - justification and overview
 - * Test tools to be used, including versions, patches, etc.
 - * Test script/test code maintenance processes and version control

Testing Interview questions and answers

- * Problem tracking and resolution - tools and processes
- * Project test metrics to be used
- * Reporting requirements and testing deliverables
- * Software entrance and exit criteria
- * Initial sanity testing period and criteria
- * Test suspension and restart criteria
- * Personnel allocation
- * Personnel pre-training needs
- * Test site/location
- * Outside test organizations to be utilized and their purpose, responsibilities, deliverables, contact persons, and coordination issues.
- * Relevant proprietary, classified, security, and licensing issues.
- * Open issues
- * Appendix - glossary, acronyms, etc.

What's a 'test case'?

- * A test case is a document that describes an input, action, or event and an expected response, to determine if a feature of an application is working correctly. A test case should contain particulars such as test case identifier, test case name, objective, test conditions/setup, input data requirements, steps, and expected results.
- * Note that the process of developing test cases can help find problems in the requirements or design of an application, since it requires completely thinking through the operation of the application. For this reason, it's useful to prepare test cases early in the development cycle if possible.

What should be done after a bug is found?

- * The bug needs to be communicated and assigned to developers that can fix it. After the problem is resolved, fixes should be re-tested, and determinations made regarding requirements for regression testing to check that fixes didn't create problems elsewhere. If a problem-tracking system is in place, it should encapsulate these processes. A variety of commercial problem-tracking/management software tools are available (see the 'Tools' section for web resources with listings of such tools). The following are items to consider in the tracking process:
- * Complete information such that developers can understand the bug, get an

Testing Interview questions and answers

idea of it's severity, and reproduce it if necessary.

- * Bug identifier (number, ID, etc.)
- * Current bug status (e.g., 'Released for Retest', 'New', etc.)
- * The application name or identifier and version
- * The function, module, feature, object, screen, etc. where the bug occurred
- * Environment specifics, system, platform, relevant hardware specifics
- * Test case name/number/identifier
- * One-line bug description
- * Full bug description
- * Description of steps needed to reproduce the bug if not covered by a test case or if the developer doesn't have easy access to the test case/test script/test tool
- * Names and/or descriptions of file/data/messages/etc. used in test
- * File excerpts/error messages/log file excerpts/screen shots/test tool logs that would be helpful in finding the cause of the problem
- * Severity estimate (a 5-level range such as 1-5 or 'critical'-to-'low' is common)
- * Was the bug reproducible?
- * Tester name
- * Test date
- * Bug reporting date
- * Name of developer/group/organization the problem is assigned to
- * Description of problem cause
- * Description of fix
- * Code section/file/module/class/method that was fixed
- * Date of fix
- * Application version that contains the fix
- * Tester responsible for retest
- * Retest date

Testing Interview questions and answers

- * Retest results
- * Regression testing requirements
- * Tester responsible for regression tests
- * Regression testing results
- * A reporting or tracking process should enable notification of appropriate personnel at various stages. For instance, testers need to know when retesting is needed, developers need to know when bugs are found and how to get the needed information, and reporting/summary capabilities are needed for managers.

What if the software is so buggy it can't really be tested at all?

* The best bet in this situation is for the testers to go through the process of reporting whatever bugs or blocking-type problems initially show up, with the focus being on critical bugs. Since this type of problem can severely affect schedules, and indicates deeper problems in the software development process (such as insufficient unit testing or insufficient integration testing, poor design, improper build or release procedures, etc.) managers should be notified, and provided with some documentation as evidence of the problem.

How can it be known when to stop testing?

This can be difficult to determine. Many modern software applications are so complex, and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are:

- * Deadlines (release deadlines, testing deadlines, etc.)
- * Test cases completed with certain percentage passed
- * Test budget depleted
- * Coverage of code/functionality/requirements reaches a specified point
- * Bug rate falls below a certain level
- * Beta or alpha testing period ends

What if there isn't enough time for thorough testing?

* Use risk analysis to determine where testing should be focused. Since it's rarely possible to test every possible aspect of an application, every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to most software development projects. This requires judgement skills, common sense, and experience. (If warranted, formal methods are also available.) Considerations can include:

Testing Interview questions and answers

- * Which functionality is most important to the project's intended purpose?
- * Which functionality is most visible to the user?
- * Which functionality has the largest safety impact?
- * Which functionality has the largest financial impact on users?
- * Which aspects of the application are most important to the customer?
- * Which aspects of the application can be tested early in the development cycle?
- * Which parts of the code are most complex, and thus most subject to errors?
- * Which parts of the application were developed in rush or panic mode?
- * Which aspects of similar/related previous projects caused problems?
- * Which aspects of similar/related previous projects had large maintenance expenses?
- * Which parts of the requirements and design are unclear or poorly thought out?
- * What do the developers think are the highest-risk aspects of the application?
- * What kinds of problems would cause the worst publicity?
- * What kinds of problems would cause the most customer service complaints?
- * What kinds of tests could easily cover multiple functionalities?
- * Which tests will have the best high-risk-coverage to time-required ratio?

What if the project isn't big enough to justify extensive testing?

* Consider the impact of project errors, not the size of the project. However, if extensive testing is still not justified, risk analysis is again needed and the same considerations as described previously in 'What if there isn't enough time for thorough testing?' apply. The tester might then do ad hoc testing, or write up a limited test plan based on the risk analysis.

What can be done if requirements are changing continuously?

A common problem and a major headache

* Work with the project's stakeholders early on to understand how requirements might change so that alternate test plans and strategies can be worked out in advance, if possible.

Testing Interview questions and answers

- * It's helpful if the application's initial design allows for some adaptability so that later changes do not require redoing the application from scratch.
- * If the code is well-commented and well-documented this makes changes easier for the developers.
- * Use rapid prototyping whenever possible to help customers feel sure of their requirements and minimize changes.
- * The project's initial schedule should allow for some extra time commensurate with the possibility of changes.
- * Try to move new requirements to a 'Phase 2' version of an application, while using the original requirements for the 'Phase 1' version.
- * Negotiate to allow only easily-implemented new requirements into the project, while moving more difficult new requirements into future versions of the application.
- * Be sure that customers and management understand the scheduling impacts, inherent risks, and costs of significant requirements changes. Then let management or the customers (not the developers or testers) decide if the changes are warranted - after all, that's their job.
- * Balance the effort put into setting up automated testing with the expected effort required to re-do them to deal with changes.
- * Try to design some flexibility into automated test scripts.
- * Focus initial automated testing on application aspects that are most likely to remain unchanged.
- * Devote appropriate effort to risk analysis of changes to minimize regression testing needs.
- * Design some flexibility into test cases (this is not easily done; the best bet might be to minimize the detail in the test cases, or set up only higher-level generic-type test plans)
- * Focus less on detailed test plans and test cases and more on ad hoc testing (with an understanding of the added risk that this entails).
- **What if the application has functionality that wasn't in the requirements?**
 - * It may take serious effort to determine if an application has significant unexpected or hidden functionality, and it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed, as it may have unknown impacts or dependencies that were not taken into account by the designer or the customer. If not removed, design information will be needed to determine added testing needs or regression testing needs. Management should be

Testing Interview questions and answers

made aware of any significant added risks as a result of the unexpected functionality. If the functionality only effects areas such as minor improvements in the user interface, for example, it may not be a significant risk.

How can QA processes be implemented without stifling productivity?

* By implementing QA processes slowly over time, using consensus to reach agreement on processes, and adjusting and experimenting as an organization grows and matures, productivity will be improved instead of stifled. Problem prevention will lessen the need for problem detection, panics and burn-out will decrease, and there will be improved focus and less wasted effort. At the same time, attempts should be made to keep processes simple and efficient, minimize paperwork, promote computer-based processes and automated tracking and reporting, minimize time required in meetings, and promote training as part of the QA process. However, no one - especially talented technical types - likes rules or bureaucracy, and in the short run things may slow down a bit. A typical scenario would be that more days of planning and development will be needed, but less time will be required for late-night bug-fixing and calming of irate customers. (See the Books section's 'Software QA', 'Software Engineering', and 'Project Management' categories for useful books with more information.)

What if an organization is growing so fast that fixed QA processes are impossible

* This is a common problem in the software industry, especially in new technology areas. There is no easy solution in this situation, other than:

* Hire good people

* Management should 'ruthlessly prioritize' quality issues and maintain focus on the customer

* Everyone in the organization should be clear on what 'quality' means to the customer

How does a client/server environment affect testing?

* Client/server applications can be quite complex due to the multiple dependencies among clients, data communications, hardware, and servers. Thus testing requirements can be extensive. When time is limited (as it usually is) the focus should be on integration and system testing. Additionally, load/stress/performance testing may be useful in determining client/server application limitations and capabilities. There are commercial tools to assist with such testing. (See the 'Tools' section for web resources with listings that include these kinds of test tools.)

How can World Wide Web sites be tested?

* Web sites are essentially client/server applications - with web servers and 'browser' clients. Consideration should be given to the interactions between

Testing Interview questions and answers

- html pages, TCP/IP communications, Internet connections, firewalls, applications that run in web pages (such as applets, javascript, plug-in applications), and applications that run on the server side (such as cgi scripts, database interfaces, logging applications, dynamic page generators, asp, etc.). Additionally, there are a wide variety of servers and browsers, various versions of each, small but sometimes significant differences between them, variations in connection speeds, rapidly changing technologies, and multiple standards and protocols. The end result is that
- testing for web sites can become a major ongoing effort. Other considerations might include:

How is testing affected by object-oriented designs?

- * What are the expected loads on the server (e.g., number of hits per unit time?), and what kind of performance is required under such loads (such as web server response time, database query response times). What kinds of tools will be needed for performance testing (such as web load testing tools, other tools already in house that can be adapted, web robot downloading tools, etc.)?
- * Who is the target audience? What kind of browsers will they be using? What kind of connection speeds will they be using? Are they intra- organization (thus with likely high connection speeds and similar browsers) or Internet-wide (thus with a wide variety of connection speeds and browser types)?
- * What kind of performance is expected on the client side (e.g., how fast should pages appear, how fast should animations, applets, etc. load and run)?
- * Will down time for server and content maintenance/upgrades be allowed? how much?
- * Will down time for server and content maintenance/upgrades be allowed? how much?
- * How reliable are the site's Internet connections required to be? And how does that affect backup system or redundant connection requirements and testing?
- * What processes will be required to manage updates to the web site's content, and what are the requirements for maintaining, tracking, and controlling page content, graphics, links, etc.?
- * Which HTML specification will be adhered to? How strictly? What variations will be allowed for targeted browsers?
- * Will there be any standards or requirements for page appearance and/or graphics throughout a site or parts of a site?
- * How will internal and external links be validated and updated? how often?
- * Can testing be done on the production system, or will a separate test system be required? How are browser caching, variations in browser option settings, dial-up connection variabilities, and real-world internet 'traffic

Testing Interview questions and answers

congestion' problems to be accounted for in testing?

- * How extensive or customized are the server logging and reporting requirements; are they considered an integral part of the system and do they require testing?

- * How are cgi programs, applets, javascripts, ActiveX components, etc. to be maintained, tracked, controlled, and tested?

- * Pages should be 3-5 screens max unless content is tightly focused on a single topic. If larger, provide internal links within the page.

- * The page layouts and design elements should be consistent throughout a site, so that it's clear to the user that they're still within a site.

- * Pages should be as browser-independent as possible, or pages should be provided or generated based on the browser-type.

- * All pages should have links external to the page; there should be no dead-end pages.

- * The page owner, revision date, and a link to a contact person or organization should be included on each page.

What is Extreme Programming and what's it got to do with testing?

- * Extreme Programming (XP) is a software development approach for small teams on risk-prone projects with unstable requirements. It was created by Kent Beck who described the approach in his book 'Extreme Programming Explained' (See the Softwareqatest.com Books page.). Testing ('extreme testing') is a core aspect of Extreme Programming. Programmers are expected to write unit and functional test code first - before the application is developed. Test code is under source control along with the rest of the code. Customers are expected to be an integral part of the project team and to help develop scenarios for acceptance/black box testing. Acceptance tests are preferably automated, and are modified and rerun for each of the frequent development iterations. QA and test personnel are also required to be an integral part of the project team. Detailed requirements documentation is not used, and frequent re-scheduling, re-estimating, and re-prioritizing is expected.