**2. How you derived Test Cases?**

That depends on project, sometimes we derived Test cases from requirements and sometimes from use cases.

**3. How many Test cases did you write for last project?**

Nearly 170 Test cases.

**4. How much time is required to write a Test case?**

That depends on complexity of the functionality.

**5. How many defects did you detect in your last project?**

I executed nearly 150 Test cases, in which some 22 defects were raised including 3 showstoppers.

**6. Did you face any problems during defect reporting and tracking?**

Yes, Developers rejected 2 or 3 valid defects.

**7. Did you create RTM (Requirements Traceability Matrix) document in your project?**

I only updated the RTM document.

**8. Did you involve in Test Environment setup?**

Yes, I involved in verifying Test Lab setup along with other team members.

**9. Did you perform any Live testing?**

Yes, In my current project we used live data for some test cases.

**10. What is the difference between Front End Testing and Back End testing?**

Front End Testing is performed on the Graphical User Interface (GUI), whereas Back End Testing involves databases testing.

We conduct Database Testing using SQL Queries.

Database Testing is a subset of Functional Testing.

**11. What is the difference between System Testing and Functional Testing?**

System Testing is a Test Level

Functional Testing is a Test Type that can be performed in all levels of Testing (Unit Testing, Integration Testing, system Testing and Acceptance Testing).

**12. What is the difference between Performance Testing and Load Testing?**

Load Testing, Stress Testing, Spike Testing and Endurance Testing all are subsets of performance Testing.

**13. What are the Test types that you performed in your Software Testing career?**

Functionality Testing

Security Testing

Usability Testing

Compatibility Testing

Installation Testing etc…

**14. What are the Test deliverables?**

What documentation we produce during testing all come under Test deliverables.

Test Plan

Test Scenarios

Test Cases

Opened and Closed Defect Reports

Test metrics reports

Test summary Report etc…

**15. Did you involve in Test plan documentation?**

Yes, I was involved in Test plan documentation in the last project, identified Features to be Tested, Entry criteria, Exit criteria.

**16. What is Exhaustive Testing?**

Exhaustive Testing – testing with all possible inputs and pre-conditions and it is impractical, so we use Test design techniques to reduce the size of Input and Output domains.

**17. What are the important phases in the Formal Software test process or Software test life cycle?**

i. Requirement Analysis  
ii. Test Planning  
iii. Test Design & Development  
iv. Test Environment Setup  
v. Test Execution  
vi. Test Cycle Closure

**18. What are the important tasks in the Test Planning phase?**

Important tasks in the Test planning phase are:

i) Understanding and Analyzing the Requirements  
ii) Risk Analysis  
iii) Test Strategy Implementation  
iv) Test Estimations (Scope, Time, Resources, Budget, etc…)  
v) Team Formation  
vi) Test Plan Documentation  
vii) Configuration Management Planning  
viii) Traceability Matrix documentation  
ix) Define Test Environment Setup

**19. What are important tasks in the Test Design phase?**

i) Understanding Requirements  
ii) Generate Test Scenarios  
iii) Test Case Documentation  
iv) Test Data Collection

**20. What are the Test design techniques that you used?**

I used Black box Test Design Techniques,

i) Equivalence Class Partitioning

ii) Boundary Value Analysis

iii) Decision Tables

iv) State Transition Testing etc…

**21. How you communicate with Developers to resolve issues?**

That depends on Company and sometimes depends on Project, in my current project I am communicating with Developers via our Test Lead.

**22. What is Configuration Management? Did you any Configuration Management Tool in your Testing career?**

Storing and Organizing all configurable items is called Configuration Management, It is not only for Testing Team but also for all Stakeholders of the Project.

It is very important for the Development than Testing Team, I used VSS Tool for Configuration Management in my last project.

**23. You told Configurable items, What are Configurable items in Software Test Process?**

What Software & Hardware we use and What Documents (Test Plan, Test Cases, defect Reports, Test Summary Report etc…) we produce during Testing all are come under Configurable items.

**24. When we choose Informal Testing?**

Whenever we don’t have proper documentation (Requirements etc…) and sufficient Time then we choose Informal Testing. Using Experienced based Techniques (Ex: Error Guessing, Exploratory Testing etc…) we conduct Testing.

**25. What are the important Test Types that can be applied for Web Applications?**

Test Types that applied for Web Applications are,

i) Functionality Testing  
ii) Security Testing  
iii) Compatibility Testing (OS Compatibility and Browser Compatibility)  
iv) Navigation Testing  
v) Database Testing  
vi) Reliability Testing  
vii) Usability Testing  
viii) Recovery Testing  
ix) Performance Testing Etc…

**26. Do You have experience in Database Testing?**

##### ****28. What is a Test Strategy and what does it include?****

It is a document that captures the approach on how we go about testing the product and achieve the goals. It is normally derived from the Business Requirement Specification (BRS). Documents like Test Plan are prepared by keeping this document as a base.

##### ****29. Have you written Test Plan?****

Yes, I wrote the Test plan for my current project.

##### ****30. What are the various tools you have used in the Testing process?****

The tools which I have used during the testing process are as follows.

Test Management Tools: JIRA, ALM/Quality Center

Test Case Management Tools: TestCaseLab

Defect Tracking Tools: Bugzilla, Mantis

### **1. What is manual testing?**

[Manual testing](https://katalon.com/resources-center/blog/manual-testing) refers to the process of manually executing test cases to identify defects in applications, without the use of automated tools. It involves human testers interacting with the system like a real user would and analyzing the results to ensure that the application functions as intended.

### **Q2. What is automated testing?**

Automated testing, in contrast with manual testing, uses frameworks and tools to automatically run a suite of test cases. The whole process from test creation to execution is done with little human intervention, helping to reduce manual effort while increasing testing accuracy and efficiency.

**Read more:**[What Is Automation Testing? Ultimate Guide & Best Practices](https://katalon.com/resources-center/blog/what-is-automation-testing)

### **Q3. Compare manual testing with automation testing**

Simply put, for manual testing, the QA team must interact with the system manually, while for automation testing, all of those interactions are performed automatically by test scripts, and testers only have to do the execution and analysis. The table below provides a more detailed comparison between manual testing and automated testing

|  |  |  |
| --- | --- | --- |
| **Aspects** | **Manual Testing** | **Automation Testing** |
| **Definition** | Software tested manually by humans without automation tools or scripts | Software tested using automation tools or scripts written by humans |
| **Human Intervention** | Requires significant human intervention and manual effort | Requires less human intervention |
| **Speed** | Slower | Faster |
| **Reliability** | More prone to human error | More reliable as it eliminates human error |
| **Reusability** | Test cases cannot be easily reused | Test cases can be easily reused |
| **Cost** | Can be expensive due to human resources required | Can be expensive upfront due to automation tools setup but cheaper in the long run |
| **Scope** | Limited scope due to time and effort limitations | Wider scope as more tests can be executed in a shorter time |
| **Complexity** | Unable to handle complex tests that require multiple iterations | Able to handle complex tests that require multiple iterations |
| **Accuracy** | Depends on the skills and experience of the tester | More accurate as it eliminates human error and follows predetermined rules |
| **Maintenance** | Easy to maintain since it does not involve complex scripts | Requires ongoing maintenance and updates to scripts and tools |
| **Skillset** | Requires skilled and experienced testers | Requires skilled automation engineers or developers |

***Read more****:*[*Manual Testing vs Automation Testing: A Full Comparison*](https://katalon.com/resources-center/blog/manual-testing-vs-automation-testing)

### **Q4. What Are Some Manual Testing Types?**

**Ad hoc testing** is a spontaneous and unplanned manual testing approach where testers perform testing without following any specific predefined test plans. Testers rely on their experience and intuition to identify (sometimes even *guess*) potential defects in the software.

[**Exploratory testing**](https://katalon.com/resources-center/blog/exploratory-testing) is another manual testing type, and although it still emphasizes spontaneity, exploratory is more systematic compared to ad hoc testing. It also focuses on learning and investigating the software under test while simultaneously designing and executing test cases on the fly.

**Usability testing**is a popular manual testing type where testers focus on assessing the ease of use, user interface, and user experience of the software. They have to put themselves in the shoes of the user and interact with the system without the help of any specialized testing tool. It adds a touch of humanity to automation testing, discovering bugs that could have been missed.

### **Q5. When Do We Need Manual Testing and Automation Testing?**

Manual testing requires human input and creativity, especially in exploratory testing where testers interact with the software to identify areas for further testing. Usability testing also requires human perspective to evaluate user experience. Manual testing allows for quick adaptation to changing requirements. 

It also has a lower learning curve and can handle complex scenarios that are difficult to automate. Not just that, manual testing is a suitable approach for small projects due to lower upfront costs, while automation testing requires investment into tools and expertise.

On the other hand, automated testing is particularly valuable for [regression testing and retesting](https://katalon.com/resources-center/blog/regression-testing), which involves running the same test cases repeatedly after software changes or updates. It is also beneficial for cross-browser and cross-environment testing, where tests need to be conducted on various browsers, devices, and operating systems, and it’s time consuming to manually test on all of them.

### **Q6. What are the different parts of a test automation framework?**

* IDE: IntelliJ, Eclipse or whichever supports the coding language you’re using
* Test libraries of functions: create utilities for writing, running, debugging and reporting automated tests with names like Selenium, JUnit, TestNG, Playwright, Appium, Rest Assured
* Project and test artifact management structures: object repositories, helper utilities
* Browser drivers
* Test design patterns and automation approach  (e.g., Page Object Model, Screenplay, Fluent)
* Coding standards (KISS, DRY, camelCasing)
* Test reports and execution logs: plugin

### **Q7. How do you choose a tool/framework for automated testing?**

Some criteria when it comes to tool selection for test automation:

* **Functionality:** the tool should have the necessary functionalities to create, run, report, and debug tests. Additionally, assess whether the tool’s strength (e.g., API testing) matches your testing needs most.
* **Usability:** the automated testing tool should have an easy-to-navigate UI with clear instructions to help you perform your tests effectively.
* **Scalability:** the tool should be scalable to meet the demands of your testing needs, both now and in the future, as your software evolves and grows.
* **Integration:**a good automation testing tool should be able to integrate with other tools in your system, such as CI/CD, bug tracking, or ALM, and test management to help streamline the testing process.
* **Support:** the tool should have good customer support and a vibrant community, with resources such as forums, online tutorials, and knowledge bases.
* **Security:** the tool should have adequate security measures in place to protect your data and ensure that your tests are performed securely.
* **Reputation:** The tool should have a good reputation in the testing community, with positive reviews and recommendations from other users and experts.

### **Q8. Explain The Software Development Life Cycle**

The **original**software development life cycle consists of 7 stages:

1. Planning

2. Analysis

3. Design

4. Development

5. Testing

6. Deployment

7. Maintenance

During planning, project scope and requirements are defined in detail. Analysis involves gathering requirements and creating specifications, then the Design stage transforms requirements into an implementable architecture.

Development involves coding, testing, and integration. Testing confirms that the software meets requirements, then, at the final stage, Deployment is when software is brought to the production environment. Maintenance in fact is an ongoing effort to ensure functionality and make updates to keep up with the changing requirements.

### **Q9. What is Shift Left Testing? Why is Shift Left Testing important?**

[Shift left testing](https://katalon.com/resources-center/blog/shift-left-testing-approach) is a software testing approach that places strong emphasis on conducting testing activities earlier in the development process by shifting all testing activities to earlier development stages rather than leaving until the very final stages. 

Shift left testing was created due to the drawbacks of waiting until after development to start testing. The testing team does not have enough time to thoroughly check the build, and even if they do, the development team will not have enough time to troubleshoot and fix the bugs, leading to delayed releases. By shifting it to the left, both teams get to have more time to do their job better, improving the project’s agility.

**Read More:**[What isShift Right Testing? Shift left vs Shift Right](https://katalon.com/resources-center/blog/shift-right-testing)

### **Q10. Explain the Software Testing Life Cycle**

**Requirements analysis**

Testers and stakeholders will analyze software requirements and align with each other on the  objectives, scopes, and schedule of the testing process. This information is consolidated into a test plan, which lays out the strategy for the following phases.

For better tracking and management, teams usually use Requirement Traceability Matrix (RTM) to map out and trace client requirements with test cases. 

**Test planning**

In this phase, [a comprehensive test plan](https://katalon.com/resources-center/blog/test-plan) is developed. It outlines the test deliverables, resource estimates, limitations, and risks. The selection of testing techniques and tools is also detailed. Teams might also create a contingency plan that includes mitigation strategies or backup test scenarios to ensure the deliverables in case of unforeseen challenges.

**Test case development**

Teams start to design test cases based on the identified requirements. This can be performed either manually or automatically with software testing tools (Katalon, LambdaTest, Ranorex) or frameworks (Cucumber, Cypress, or Selenium).    

*Read more:*[*Top 15 Automation Testing Tools | Latest Update in 202*](https://katalon.com/resources-center/blog/automation-testing-tools)*4*

#### **Environment setup**

It involves setting up the hardware, software, network configurations, and any additional tools or test data required for testing. The test environment should closely resemble the production environment to ensure accurate testing.

Teams can choose between cloud environments or physical devices. The final decision depends on the team’s resources, priorities, and the nature of the AUT.

#### **Test execution**

Testers input the test data, observe the system's behavior, and compare the actual results with the expected results. Defects or issues discovered during testing are reported and tracked for resolution.

*Read more about software testing:*[*What is Software Testing? Definition, Types, and Tools*](https://katalon.com/resources-center/blog/software-testing)

#### **Test cycle closure**

In the final stage, testers evaluate the test results and generate test reports summarizing the test coverage, defects found, and their resolution status. QA teams and relevant stakeholders also discuss the overall test cycle, lessons learned, and identify areas for improvement in future testing cycles.

Read: [[Free E-Book] Questions to Improve Software Testing Processes](https://katalon.com/agile-qa-lead-questions)

### **Q11. What are the differences between a bug, a defect, and an error?**

**A bug** is a coding error in the software that causes it to malfunction during testing. It can result in functional issues, crashes, or performance problems.

**A defect** is a discrepancy between the expected and actual results that is discovered by the developer after the product is released. It refers to issues with the software's behavior or internal features.

**An error** is a mistake or misconception made by a software developer, such as misunderstanding a design notation or typing a variable name incorrectly. Errors can lead to changes in the program's functionality.

### **Q12. Explain the Bug Life Cycle in detail.**

The bug life cycle is the stages that a bug goes through from the time it is discovered until the time it is resolved or closed. The bug life cycle typically includes the following stages:

1. **New**: The bug is reported by the tester or user and is in the initial stage. At this stage, the bug is not yet verified by the development team.
2. **Assigned**: The bug report is assigned to a developer or development team for review and analysis. The developer reviews the bug report and determines if it is a valid issue.
3. **Open**: The developer confirms that the bug report is valid and assigns it the status of "Open". The developer then begins working on a fix for the bug.
4. **In Progress**: The developer starts working on fixing the bug. The status of the bug is changed to "In Progress".
5. **Fixed**: After the developer fixes the bug, the status of the bug is changed to "Fixed". The developer then passes the bug to the testing team for verification.
6. **Verified**: The testing team tests the bug fix to ensure that it has been properly resolved. If the bug fix is verified, the bug status is changed to "Verified" and is sent back to the development team for deployment.
7. **Closed**: Once the bug has been deployed to the production environment and verified by the users, the bug is marked as "Closed". The bug is now considered to be resolved.
8. **Reopened**: If the bug resurfaces after being marked as "Closed", it is assigned a status of "Reopened" and sent back to the development team for further analysis and resolution.

**Read More:** [Bug Life Cycle in Software Testing](https://katalon.com/resources-center/blog/the-anatomy-of-a-bug-lifecycle-and-some-tools-to-improve-it)

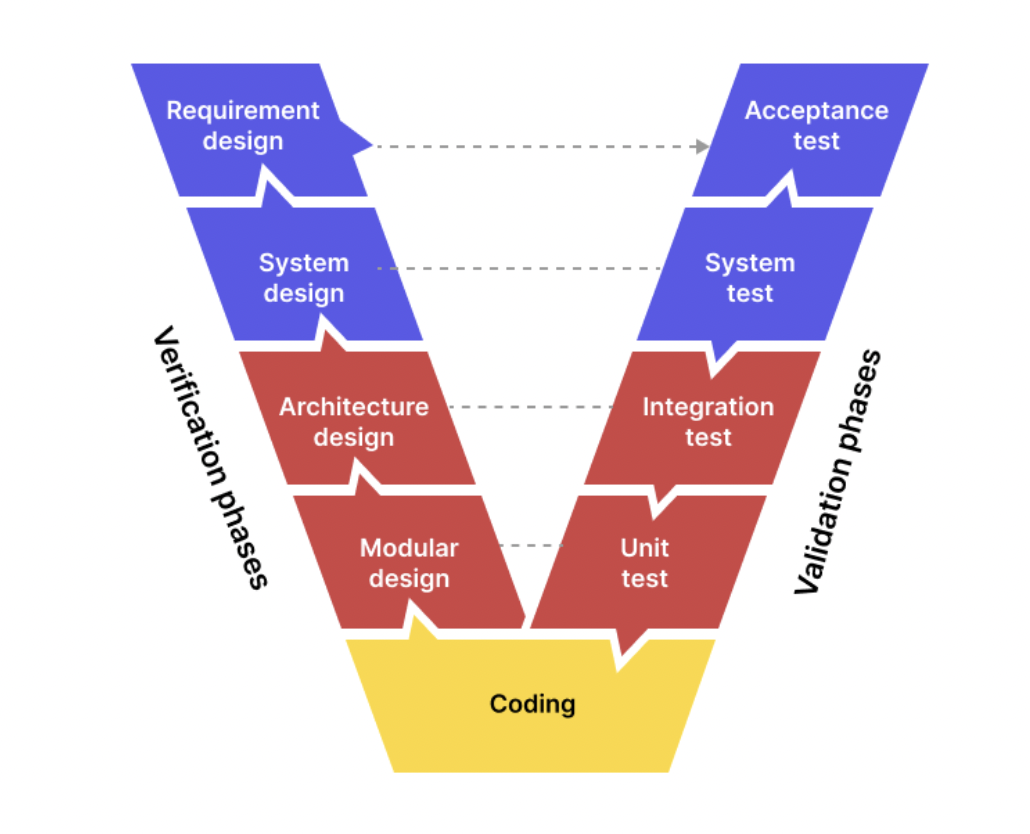
### **Q13. Differences between Waterfall Model, V-Model, and Agile Model.**

**Waterfall Model**

The Waterfall model is a sequential, linear approach to software development. It consists of a series of phases such as requirements gathering, design, implementation, testing, deployment, and maintenance, where each phase must be completed before moving on to the next. The Waterfall model is characterized by its rigid and inflexible structure, which makes it difficult to incorporate changes once a phase is completed.

**V-Model**

The V-Model is an extension of the Waterfall model that emphasizes the relationship between testing and development. In the V-Model, the testing activities are planned and executed in parallel with the development activities. Each stage of the development process has a corresponding testing stage that validates the outputs of the previous stage. This approach helps to identify defects early in the development cycle, reducing the cost of fixing them later on.



**Agile Model**

Agile is a flexible and iterative approach to software development that emphasizes collaboration, continuous delivery, and rapid feedback. Agile teams work in short development cycles, called sprints, and prioritize working software over comprehensive documentation. Agile methodologies such as Scrum, Kanban, and Extreme Programming (XP) focus on continuous improvement and adaptability, enabling teams to respond quickly to changing requirements.

### **Q14. What Are Test Cases And How Do You Write Test Cases?**

A test case is the set of actions required to validate a particular software feature or functionality. In software testing, a test case is a detailed document of specifications, input, steps, conditions, and expected outputs regarding the execution of a software test on the AUT.

A standard test case typically contains:

* Test Case ID
* Test Scenario
* Test Steps
* Prerequisites
* Test Data
* Expected Results
* Actual Results
* Test Status

Testers can create a spreadsheet for test cases with a column for each of the information above. There is a [Google Sheet template](https://docs.google.com/spreadsheets/d/1DhgK9fWPGIxaB86RdUoSrDc0LWUZO4iMTdCkvv0E8Bw/edit) available for this purpose that they can customize based on their needs. However, for more advanced test case management, they may need some [test management tools](https://katalon.com/testops) to support them.

### **Q15. What Are Some Best Practices In Writing Test Cases?**

* **Use clear and concise language:** Too abstract description, not enough details, or too many details are common mistakes that should be avoided. For example, instead of using the general “Check the purchase functionality", you can write “Check adding a product to cart".
* **Maximize test coverage:**Ensure different aspects of the software are tested thoroughly with all possible scenarios and edge cases. Use the Requirements Traceability Matrix (RTM) that maps and traces user requirements to ensure all of them are met.
* **Data-driven testing:**Test data should be realistic and representative of real-world scenarios. For example, if you test a login page, data such as names, email addresses, and passwords should be diverse to cover both positive and negative cases.
* **Review and update:**Code changes can break existing tests. Tests should be maintained periodically to keep test artifacts up to date with new feature releases, upgrades, and bug fixes.

### **Q16.What is regression testing? Why is it necessary?**

[Regression testing](https://katalon.com/resources-center/blog/regression-testing) is a software testing practice that involves re-executing a set of software tests whenever code changes happen. By introducing new code changes, the existing code may be affected, which could lead to defects or malfunctions in the software. To minimize potential risks, regression testing is implemented to ensure that the previously developed and tested code remains operational when new features or code changes are introduced.

### **Q17.Can regression testing be done manually?**

Yes, regression testing can be done manually. However, when the number of regression test cases increases along with the size of the project, manually executing all test cases is extremely time-consuming and counter-productive. It is better to automate regression testing at this point.

Even when testers choose automated regression testing, they still need to be selective, since not all test cases can be automated. A good rule of thumb is that repetitive and predictable test cases should be automated, while one-off, ad-hoc, or non-repetitive tests should not.

[](https://katalon.com/shift-from-manual-automated-testing)

## **Advanced Manual Testing Interview Questions**

In this section, we’ll explore nuances of testing types and

### **Q18. What is Smoke Testing?**

Smoke testing is a non-exhaustive testing type that ensures the proper function of basic and critical components of a single build. It is performed in the initial phase of the software development life cycle (SDLC) on attaining a fresh build from developers.

### **Q19. What is Sanity Testing? How different is Sanity Testing from Smoke Testing?**

Sanity testing is performed to ascertain no issues arise in the new functionalities, code changes, and bug fixes. Instead of examining the entire system, it focuses on narrower areas of the added functionalities. The main objective of smoke testing is to test the stability of the critical build, while that of sanity testing is to verify the rationality of new module additions or code changes.

Below is a table to compare sanity testing with smoke testing:

|  |  |
| --- | --- |
| **Smoke testing** | **Sanity testing** |
| Executed on initial/unstable builds | Performed on stable builds |
| Verifies the very basic features | Verifies that the bugs have been fixed in the received build and no further issues are introduced |
| Verify if the software works at all | Verify several specific modules, or the modules impacted by code change |
| Can be carried out by both testers and developers | Carried out by testers |
| A subset of acceptance testing | A subset of regression testing |
| Done when there is a new build | Done after several changes have been made to the previous build |

### **Q20. Can Sanity Testing Be Automated?**

Automation for sanity testing should be used with proper judgment. Since sanity testing focuses on a limited set of critical functionalities, it may not be practical to automate all aspects of this testing type.

**Read more:**[Sanity Testing vs Smoke Testing: A Comparison](https://katalon.com/resources-center/blog/sanity-testing-vs-smoke-testing)

### **Q21. What is Unit testing? Who performs unit testing?**

[Unit testing](https://katalon.com/resources-center/blog/unit-testing) is a software testing technique where individual components of an application are tested in isolation from the rest of the application to ensure that they work as intended. It is typically **performed by developers** during the development phase to detect and fix defects early in the development cycle.

### **Q22. What are the differences between regression testing and retesting?**

**Retesting** literally means “test again” for a specific reason. Retesting takes place when a defect in the source code is fixed or when a particular test case fails in the final execution and needs to be re-run. It is done to confirm that the defect has actually been fixed and that no new bug surfaces from it. 

**Regression testing** is performed to find out whether the updates or changes had caused new defects in the existing functions. This step would ensure the unification of the software.

Retesting solely focuses on the failed test cases while regression testing is applied to those that have passed, in order to check for unexpected new bugs. Another important note is that retesting includes error verifications, in contrast to regression testing, which includes error localization.

### **Q23. How to perform system testing and why is it important?**

System testing is a type of software testing that aims to test the entire system or application as a whole, rather than individual components or modules. The goal of system testing is to ensure that the system or application is functioning correctly, meeting all requirements, and performing as expected in its intended environment.

By testing the system as a whole, system testing can reveal defects that arise from the interactions between different components, modules, or systems, and can uncover issues related to performance, usability, security, and other non-functional requirements. System testing provides an opportunity to assess the overall quality and readiness of the system for deployment and can help to mitigate the risks associated with deploying a faulty or unreliable system.

Furthermore, system testing can provide valuable feedback to developers and stakeholders, enabling them to identify areas for improvement and refine the requirements or design of the system. It can also provide assurance to customers and other stakeholders that the system has been thoroughly tested and is ready for use.

### **Q24. Give some examples of high-priority and low-severity defects.**

Defects can be prioritized based on their severity and impact on the software application. Let's look at an example of an ecommerce website. 

High-priority defects:

* Checkout process failure, preventing users from completing transactions.
* The payment processing system is not working correctly, causing errors or failed transactions.
* User's personal information is not properly secured or is vulnerable to hacking.
* The inventory management system is not working correctly, resulting in incorrect product availability information.
* Search functionality is not working correctly, making it difficult for users to find products they are looking for.

Low-severity defects:

* Minor formatting or styling issues on product or checkout pages.
* Navigation issues, such as links not working or menus not displaying correctly.
* Spelling or grammatical errors.
* Issues that affect only a small subset of users or have a low frequency of occurrence, such as errors in product reviews or ratings.

### **Q25. What are the tools which you can use for defect tracking?**

Any list of the most popular bug management systems should include Bugzilla, JIRA, Assembla, Redmine, Trac, OnTime, and HP Quality Center. Bugzilla, Redmine, and Trac are freely available, open-source software, while Assembla and OnTime are offered in the form of software as a service. All of these options support manual bug entry, but Bugzilla, JIRA, and HP Quality Center also support some kind of API so that you can plug in third-party tools in order to enter new bug reports.

### **Q26. Explain integration testing and its different types.**

[Integration testing](https://katalon.com/resources-center/blog/integration-testing) involves testing the interfaces between the modules, verifying that data is passed correctly between the modules, and ensuring that the modules function correctly when integrated with each other.

* **Big Bang Approach**: an approach in which all modules are integrated and tested at once, as a singular entity. The integration process is not carried out until all components have been successfully built and unit tested.
* **Incremental Approac**h: opposite to Big Bang approach, the Incremental approach involves strategically selecting 2 or more modules with closely related logic to integrate and test. This process is repeated until all software modules have been integrated and tested. The advantage of this is that we can test the application at an early stage.

### **Q27. What are the differences between white-box testing and black-box testing?**

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Black Box Testing** | **White Box Testing** |
| **Definition** | A testing approach where the internal structure and logic of the system under test are not known to the tester. | A testing approach where the tester has knowledge of the internal structure, design, and implementation of the system. |
| **Focus** | External behavior and functionality of the system. | Internal logic, code coverage, and structural aspects of the system. |
| **Knowledge Required** | No knowledge of internal code or design is required. | In-depth knowledge of programming languages, code, and system architecture is needed. |
| **Test Design** | Test cases are designed based on functional requirements and user expectations. | Test cases are designed based on internal code paths, branches, and data flow within the system. |
| **Test Coverage** | Emphasizes testing from a user's perspective, covering all possible scenarios and inputs. | Can target specific code segments, paths, and branches for thorough coverage. |
| **Test Independence** | Tester and developer are independent of each other. | Collaboration between testers and developers is often necessary for effective testing. |
| **Skills Required** | Understanding of system requirements, creativity in designing test cases, and domain knowledge. | Strong programming skills, debugging capabilities, and technical expertise are required. |
| **Test Maintenance** | Tests are less affected by changes in the internal structure of the system. | Tests may need to be updated or reworked when there are changes in the code or system design. |
| **Test Validation** | Ensures that the system meets customer requirements and behaves as expected. | Verifies the correctness of the implementation, adherence to coding standards, and optimization of the system. |
| **Test Types** | Functional testing, system testing, regression testing, acceptance testing, etc. | Unit testing, integration testing, code coverage testing, security testing, etc. |
| **Advantages** | Tests are designed from a user's perspective, independent of the implementation. | Can achieve higher code coverage, identify coding errors, and optimize system performance. |
| **Disadvantages** | Limited visibility into the internal workings of the system, potential for missing edge cases or implementation issues. | Highly dependent on technical expertise, time-consuming, and may miss requirements or user expectations. |

### **Q28. What is alpha testing and beta testing?**

Alpha testing and beta testing are two types of acceptance testing that are conducted to evaluate the software application before it is released to the market.

Alpha testing is the first stage of testing and is typically conducted by the development team or a dedicated quality assurance team. It is performed in a controlled environment and is designed to catch any defects or issues before the application is released to a larger audience. During alpha testing, the application is tested in-house by the development team and feedback is provided by the team members.

Beta testing, on the other hand, is the second stage of testing and is typically conducted by a select group of external users who are invited to test the application before it is released to the market. This type of testing is also referred to as user acceptance testing or field testing. During beta testing, the application is tested in a real-world environment, and users provide feedback on the application's functionality, usability, and overall experience.

### **Q29. Explain user acceptance testing (UAT) in detail.**

Acceptance tests are functional tests that determine how acceptable the software is to the end users. UAT is typically conducted after the completion of system testing and before the software system is released to production.

After the development and testing team have agreed that everything is good to go, it’s now about finding out if stakeholders think the same. 

In software projects, a week or less will be fully dedicated to the people that wanted the product in the first place. As these users do not have a quality engineering background, they will go through the app according to instinct. Moreover, user acceptance testing isn’t for finding defects like in earlier quality stages. It instead gives a high-level view of the app and helps to determine if everything makes sense as a whole.

### **Q30. Explain load testing and stress testing. Give an example for each.**

**Load testing** sees if an app takes seconds or an eternity to respond back to user requests. Quality engineers use software load testing tools to simulate a specific workload that mimics the normal and peak number of concurrent users, then measures how much the response time is affected. 

Suppose Team A wants to see how their app behaves under traffic spikes for a holiday sale of an e-commerce website. The goal is to determine if 100,000 users rushing to find coupons would result in frustrated users staring at a spinning loading icon for over 3 minutes to make a payment.

**Stress testing** pushes a software or app beyond its limits. Whatever the average load of a system is, stress testing takes it a step further. Beyond the reliable state, software must be stress-tested for its breaking point(s) and corresponding remedies.

For example, Team A is managing a web-based application for college course enrollment. Usually, when courses open for selection every semester, there are roughly 150 concurrent users standing by when the time hits. In case the number of sessions reaches 170, the system would remain stable or recover quickly if it crashes.

### **Q31. What is the difference between UI and GUI testing?**

Both UI (User Interface) and GUI (Graphical User Interface) testing fall into functional testing, but they have different focuses. 

The main difference between UI Testing and GUI Testing lies in their scope. UI Testing focuses specifically on the user interface elements and their appearance, while GUI Testing includes testing the **both**underlying functionality and logic of the graphical components in addition to the user interface.

**Read More:** [What is UI Testing?](https://katalon.com/resources-center/blog/functional-testing)

### **Q32. Explain Monkey Testing and different types of monkey testing**

Monkey testing is a type of software testing that involves randomly generating inputs to test the behavior of a system or application. Monkey testing is to test the system under unexpected and unpredictable conditions to uncover potential errors or defects.

Several types of monkey testing are:

* **Smart monkey testing**: A more intelligent algorithm is used to generate test cases. The algorithm is designed to mimic the behavior of an experienced tester, with the aim of uncovering defects more efficiently.
* **Dumb monkey testing**: Completely random inputs are generated without any consideration for the system or application being tested. This can be useful for uncovering unexpected or edge-case defects, but can also be less efficient than smart monkey testing.
* **Syntax-based testing**: This type of testing involves generating inputs that conform to the syntax or grammar of the system or application being tested. The aim is to test the input validation and parsing mechanisms of the system.
* **Semantic testing**: This type of testing involves generating inputs that are valid but semantically incorrect, with the aim of testing the behavior of the system under unexpected conditions.
* **Hybrid testing**: This involves a combination of different monkey testing approaches, such as using smart algorithms for generating inputs and syntax-based testing for validating the inputs.

### **Q33. List out key differences between test cases and test scenarios**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Test cases** | **Test scenarios** |
| **Definition** | A detailed set of steps to execute a specific test | A high-level concept of what to test, including the environment |
| **Level of detail** | Highly specific | Broad and general |
| **Purpose** | To verify a specific functionality or requirement | To test multiple functionalities or requirements together |
| **Input parameters** | Specific test data and conditions | General data and conditions that can be applied to multiple tests |
| **Number of executions** | Single execution for each test case | Multiple executions for each test scenario, covering multiple cases |
| **Test coverage** | Narrow and focused on a single functionality | Broad and covering multiple functionalities |

### **Q34. What is API Testing? What are the types of API Testing?**

[APIs](https://katalon.com/resources-center/blog/api-examples) are the middle layer between the presentation (UI) and the database layer that allow different software applications to communicate and exchange data with each other. 

There are several types of API testing, including:

* **Unit testing**: Testing individual units or components of an API in isolation to ensure they function correctly.
* **Functional testing**: Testing the overall functionality of an API to ensure it meets the expected requirements.
* **Load testing**: Testing the performance of an API under different levels of load and stress to ensure it can handle high volumes of requests.
* **Security testing**: Testing the API's security features, such as authentication and authorization, to ensure they are effective and prevent unauthorized access.
* **Runtime or integration testing**: Testing the API in a real-world environment to ensure it integrates properly with other systems and applications.
* **Penetration testing**: Testing the API for vulnerabilities and potential exploits to ensure it is secure against malicious attacks.
* **Fuzz testing**: Testing the API with a large volume of random or invalid data to identify potential errors or security vulnerabilities.

### **Q35. What is a test environment?**

A test environment is a setup of software and hardware that is configured to execute software testing. It provides an environment for testing applications under real-world scenarios to identify and resolve issues before deployment.

The test environment is designed to mimic the production environment to ensure that the testing results are accurate and reliable. It can include servers, databases, operating systems, network configurations, and other software and hardware components needed for testing.

### **Q36. What is browser automation?**

Browser automation refers to the process of automating tasks or interactions that a user would typically perform in a web browser. It involves using software tools or frameworks to control a web browser programmatically, simulating user actions such as clicking buttons, filling out forms, navigating between pages, and extracting data.

### **Q37. What is cross-browser testing?**

[Cross-browser testing](https://katalon.com/resources-center/blog/cross-browser-testing) is the process of testing web applications or websites across multiple web browsers and their versions to ensure compatibility and consistency in functionality, design, and user experience.

### **Q38.Why do you need cross-browser testing?**

Cross-browser testing is essential because web browsers differ in their rendering engines, HTML/CSS support, JavaScript execution, and user behavior, which can lead to inconsistencies and errors in web pages or applications. Conducting cross-browser testing helps identify and fix these issues, ensuring that the web application or website works as intended across all major web browsers and platforms, providing a consistent user experience to all users.

### **Q39.What is the test automation pyramid?**

The test automation pyramid is a framework that helps to guide software testing teams in building efficient and effective test automation suites. It is composed of three layers, each representing a different level of testing:

* **Unit Testing**: The base of the pyramid, which focuses on testing individual units or components of the software. It is usually done by developers and automation engineers using testing frameworks such as JUnit or NUnit.
* **Integration Testing**: The middle layer, tests how the different components of the system work together. This is usually done using API testing, where the focus is on testing the communication and data exchange between components.
* [**End-to-End Testing**](https://katalon.com/resources-center/blog/end-to-end-e2e-testing): The top layer, which tests the entire application from the user's perspective. This is usually done using automated UI testing, where the focus is on testing the functionality and user experience of the application.

### **Q40.Who should be responsible for test automation?**

To ace this question, instead of assigning the responsibility of test automation to a specific role, your answer should be the whole team’s collaborative effort. 

While individual roles such as QA, developers, or team leads have different expertise and contribution to software quality, it is, at the end of the day, the ultimate goal of a whole project. Therefore, team members should cover each other on every aspect of operations, including test automation.

Collaboration is further emphasized in “Shift-left" testing, in which testing activities are moved earlier in the SDLC. Developers and testers work closely to identify areas for automation, define test cases, and implement test automation frameworks and tools.  This approach enables teams to achieve a faster feedback loop, continuous development, and increased overall quality upon delivery.

### **Q41.What is a test automation platform?**

A test automation platform is a comprehensive software tool or framework that provides a set of integrated features and capabilities to support the automation of testing processes. It serves as a centralized platform for designing, developing, executing, and managing automated tests.

Depending on their specific needs and resources, teams can choose to buy or build a test automation solution. If your team members are experienced developers who want to tailor and have full control of testing frameworks, Selenium or Appium is a good choice. Otherwise, a software quality management platform such as Katalon is a perfect fit if teams aim to skip the framework development and kick-start automation right away without much coding experience.

### **Q42.What are some of the alternatives to Selenium?**

Selenium is a library that contains predefined code snippets used to build common functions of a testing framework. If you seek a Selenium alternative to build open-source test frameworks from scratch, Cypress, Appium or Cucumber can be the go-to solution as they provide you with IDE, libraries of functions, coding standards, etc for framework development.

Another approach is shopping for vendor solutions, namely Katalon, LamdaTest, Postman, or Tricentis Tosca, etc. All the testing essentials -  templates, test case libraries, and keywords - are already built-in, which reduces the coding effort and allows manual QA to automate their testing.

### **Q43.What is Protractor?**

Protractor is an end-to-end testing framework specifically designed for Angular and AngularJS applications. It is a popular open-source tool used by software testers for automating tests for Angular web applications. **Please note that Protractor has** [**reached its end-of-life as of August 2023**](https://www.protractortest.org/#/project-status)**.**

Protractor simplifies the testing process for Angular applications by providing a dedicated framework with Angular-specific features and seamless integration with Angular components. It enables testers to automate end-to-end scenarios, interact with elements easily, and perform reliable and efficient testing of Angular applications.

## **Real-life Manual Testing Interview Questions**

These are all manual testing interview questions for real-life scenarios where simply remembering the theory and concepts wouldn’t be enough, and some critical thinking is necessary. It examines how well the tester can apply their technical knowledge to address real business problems and provide value.

### **Q44.What are some important aspects of the software/application that you need to test in cross browser testing?**

The QA team should compile a checklist for their cross-browser compatibility testing. Key areas to include in the test plan are:

1. Base Functionality (core features that are used the most, or generate the most value)

2. Navigation (menus, links, buttons, and any redirecting)

3. Forms and Inputs

4. Search Functionality

5. User Registration and Login

6. Web-specific Functionalities (unique eCommerce or SaaS website features)

7. Third-party Integrations (test the data exchange and communication between integrations, APIs, etc.)

8. Design (visual layout, text, font size, color)

9. [Accessibility](https://katalon.com/resources-center/blog/accessibility-testing) (whether the website is friendly for physically challenged users)

10. Responsiveness (whether the website is also responsive on mobile devices)

### **Q45. How do you ensure that the test cases cover all possible scenarios when testing?**

It may not be possible to cover every single scenario, and it is actually quite unrealistic to have a bug-free application, but testers should always aim to go beyond the **happy path** and explore additional areas. 

This means that in addition to the typical test cases, we should also focus on edge cases and negative scenarios that involve unexpected inputs or usage patterns. Including these scenarios in the test plan enhances test coverage and helps identify vulnerabilities that attackers may exploit.

### **Q46. Describe the process you follow when reporting a bug. What information do you include in a defect report?**

Testers usually follow this process to report a bug:

1. Reproduce the bug and collect essential details, including reproduction steps, screenshots, logs, and system configurations.
2. Determine the bug’s severity level based on its impact on the application and users.
3. Record the bug in a tracking tool, providing a precise description, expected outcomes, actual results, and reproduction steps.
4. Inform the development team about the bug, collaborating with them to identify the root cause and potential solutions.
5. Consistently monitor the bug’s progress until it is resolved and confirmed as fixed.

To best describe the bug, both teams usually have an agreed upon list of bug taxonomies (or bug categories) to classify and identify the type of bugs for better understanding and management. Several basic bug categories include:

1. Severity (High - Medium - Low impact to system performance/security)
2. Priority (High - Medium - Low urgency)
3. Reproducibility (Reproducible, Intermittent, Non-Reproducible, or Cannot Reproduce)
4. Root Cause (Coding Error, Design Flaw, Configuration Issue, or User Error, etc.)
5. Bug Type (Functional Bugs, Performance Issues, Usability Problems, Security Vulnerabilities, Compatibility Errors, etc.)
6. Areas of Impact
7. Frequency of Occurrence

### **Q47.How do you prioritize the test cases to execute?**

When prioritizing test cases, QA professionals consider various criteria. Here are 9 common ones:

* Business impact: Verify critical flows that impact the bottom line like Login and Checkout.
* Risk: Test high-risk areas in the application.
* Frequency of use: Test features used by a large number of users.
* Dependencies.
* Complexity.
* Customer or user feedback: Address customers' pain points.
* Compliance requirements: Cover industry-specific regulations.
* Historical data: Test cases with a history of defects or issues.
* Test case age: Update and maintain old test cases as needed.

### **Q48.When faced with limited time to test a complex software application, how would you prioritize your testing efforts and ensure sufficient coverage of critical areas? What factors would you consider in making these decisions?**

The process here would be to first **conduct exploratory testing**to understand the software application and detect the first bugs. This exploratory session will provide the tester with insights into the current state of the application and help them identify the areas that should be focused on the most in future test runs. The priority of the project would be on the test cases with highest business impact or urgency. 

After that, the dev team and the QA team will have a meeting to discuss and write a test plan, outlining all of the important variables in the project, including:

* Objectives
* Approach
* Scope
* Test Deliverables
* Dependencies
* Test Environment
* Risk Management
* Schedule
* Roles and Responsibilities

### **Other Manual Testing Interview Questions**

Other than technical and specialized questions, there are several personal interview questions that aim to discover your previous job experiences and your familiarity with tools in software testing. Examples of those questions include:

Q49. Can you provide an overview of your experience as a QA tester? How many years have you been working in this role?

Q50. Describe a challenging testing project you have worked on in the past. What were the key challenges you faced, and how did you overcome them?

Q51. Have you worked on both manual and automated testing projects? Could you share examples of projects where you utilized both approaches?

Q52. What tools and technologies have you worked with in your testing projects? Which ones are you most proficient in?

Q53. What motivated you to pursue a career in QA testing, and what keeps you interested in this field?

Q54. Describe a situation where you had to work closely with developers, product owners, or other stakeholders to resolve a testing-related issue. How did you ensure effective communication and collaboration?

Q55. Have you ever identified a significant risk or potential problem in a project? How did you handle it, and what actions did you take to mitigate the risk?

### **Download Katalon To Improve Your Testing Sk**