1. What is Automation Testing?

Automated Testing is a type of software testing that automates the process of validating the functionality of the software and ensures that the software meets its requirements before being released for production.

Automated testing is important for continuous testing, continuous integration, and continuous delivery software development practices.

It needs manual effort when creating initial scripts, and further process is performed automatically to compare the results of actual testing with expected results.

2. When will you avoid Automation Testing?

Functionality changes regularly: The software or functionality under test changes regularly thus the automated tests need to be updated frequently. Hence, the test can soon become obsolete and cease to be useful.

Inefficient for UI bugs: Automated tests unless not programmed to check for UI flaws will not find any.

Not appropriate for exploratory testing: Exploratory testing is also not appropriate for automated testing.

Tests that cannot be automated 100%: Tests that cannot be automated 100% should not be automated at all unless it saves considerable time in automating them.

Tests requiring random testing: Tests that require random testing based on domain expertise should not be automated.

3. What is a Test Environment?

The test environment is a setup of hardware and software for the testing teams to execute test cases.

The test environment also known as the test bed is configured as per the need of the Application Under Test.

The test environment setup includes system and applications, test data, database server, browser, network, etc.

4. What is Browser Automation?

Browser automation is the process of testing software in the browser using automated tools to reduce testing time, effort and deliver faster results efficiently.

5. Explain different types of Automation Tests.

Functional Testing: Functional testing focuses on assessing the software against the set of functional requirements and testing the main functions of the system.

Unit Testing: Unit testing involves testing the individual components of the system in isolation to verify that they are working as required.

Integration Testing: Integration testing involves testing all the various units of the application in unity to verify whether the system as a whole complies with the functional requirements set or not.

Smoke Testing: Smoke testing also known as confidence testing involves assessing the stability and viability of the deployed build of the software.

Non-functional Testing: Non-functional testing involves testing all the non-functional requirements of an application.

Performance Testing: Performance testing involves testing the software’s speed, responsiveness, and stability under workload.

Regression Testing: Regression testing involves rerunning of functional and non-functional tests to see if the software is performing the same way as before after some changes in the code have been made.

Keyword-driven Testing: Keyword-driven testing involves identifying the keywords and connecting them with specific actions. Every time the keyword is used, related action will be automatically done.

Data-driven Testing: Data-driven testing involves using external data that is stored in the spreadsheet format or tables to extend automated tests.

6. List factors that define the success of Automation Testing.

Some of the factors that are used to check the effectiveness of automation testing are:

Reusability

Time-saving

Quality of the software

Maintenance

Installment costs

Test coverage

The number of bugs found.

7. Who should be responsible for Test Automation?

Automation Test Engineer also known as Test Automation Developer is an Automation Tester responsible for the design, development, and testing of programs that run automated tests of hardware and software throughout the development cycle.

8. What is a Test Automation Platform?

Test Automation Platform is a system that automates the entire automation testing process by utilizing the programs and scripts written by the developers or testers. Thus, making the automated software testing simple.

9. Is it possible to achieve 100% Automation?

It is almost impossible to achieve 100% automation in any testing process due to the presence of some real-life scenarios like [captcha](https://www.geeksforgeeks.org/captcha-recaptcha-and-related-things) matching that require human intervention. [Manual testing](https://www.geeksforgeeks.org/software-testing-manual-testing) is preferred in some cases and it is not recommended to use 100% automation in any case.

10. What is the average number of test cases you have automated per day?

The average number of test cases depends upon the length and complexity of the test scenario. In general, a QA tester can automate 2-4 test scenarios daily when the complexity is limited, and sometimes only 1-2 test scenarios can be automated when the complexity is high.

11. What are the Benefits of Automation Testing?

Here are some benefits of Automatin testing:

Simplifies Test Case Execution: Automation testing can be left virtually unattended and thus it gives an opportunity to monitor the results at the end of the process.

Improves Reliability of Tests: Automation testing ensures that there is equal focus on all the areas of the testing, thus ensuring the best quality end product.

Increases amount of test coverage: Using automation testing, more test cases can be created and executed for the application under test. This allows for the testing of more complex applications and more features can be tested.

Minimizing Human Interaction: In automation testing, everything is automated from test case creation to execution thus there are no changes for human error due to neglect.

Saves Time and Money: The initial investment for automation testing is on the higher side but it is cost-efficient and time-efficient in the long run. This is due to the reduction in the amount of time required for test case creation and execution which contributes to the high quality of work.

Earlier detection of defects: Automation testing documents the defects, thus making it easier for the development team to fix the defect and give a faster output. The earlier defect is identified, the more easier and cost-efficient it is to fix the defects.

12. What are the Limitations of Automation Testing?

High initial cost: Automation testing in the initial phases requires a lot of effort in selecting the tool and designing customized software.

100% test automation is not possible: Generally, the effort is to automate all the test cases but in practical real situations not all test cases can be automated there are some test cases that require human intervention for careful observation.

Not possible to automate all testing types: It is not possible to automate tests that verify the user-friendliness of the system. Similarly, if we talk about the graphics or sound files, even their testing cannot be automated as automated tests typically use textual descriptions to verify the output.

Programming knowledge is required: Every automation testing tool uses any one of the programming languages to write test scripts. Thus, it is mandatory to have programming knowledge for automation testing.

False positives and negatives: Automation tests may sometimes fail and reflect that there is some issue in the system but actually there is no issue present and in some cases, it may generate false negatives if tests are designed to verify that some functionality exists and not to verify that it works as expected.

13. What are the prerequisites of Automation Testing?

Some of the prerequisites of automation testing are:

A stable build.

Functionalities to be tested.

Test cases for automated testing.

14. What is TestNG in Selenium?

TestNG is a powerful testing framework that provides features like flexible test configurations, test cases that can be grouped, support for data-driven testing, inbuilt reporting, etc.

It supports parameterization.

The annotations are used to create test cases easily.

It supports parallel execution.

It also supports log generation.

15. Why use Automation Frameworks?

The automation framework offers the benefit of reusing the code in various scenarios and it also helps the team to write down the test script in a standard format. With the help of automation testing frameworks, it is possible to achieve continuous testing of coding and delivery.

16. What is an Automated Test Script?

Automated test scripts are scripts where the script programmatically performs the setup and validation on its own. Automated test scripts help to save time, reduce errors, and serve as documentation for the software.

Automation Testing Interview Questions for Intermediate

Here in this section, we have filtered automated interview questions for intermediate candidates. Go through this section to get a hand on all the intermediate questions for an automated software tester.

17. Mention different parts of the Test Automation Framework.

Six major components of the test automation framework are:

System to be tested: The subsystems of the system to be tested must be stable before the start of the automation test project.

Testing platform: The testing platform on which the system will be tested must be in place to carry out the test automation project.

Test case library: The test case library is useful to compile the libraries of reusable test steps of basic utilities that are to be used as the building blocks of automated test scripts.

Automated testing practice: The procedure describing how to automate test cases using test tools and test libraries must be documented.

Testing tools: Different tools like test automation tools, traffic generation tools, and traffic monitoring tools, support tools are required for the development of test scripts.

Test administrator: The test administrator manages the test case libraries, test platforms, and test tools. It is also responsible for maintaining an inventory of templates, providing tutorials, etc.

18. Should you automate all testing?

Automated testing no doubt saves a lot of time and effort but it is impractical to automate all forms of testing. User interface testing, accessibility testing, and usability testing are some forms of testing that cannot be automated and can only be carried out by human testers.

19. What is Cross-browser Testing?

Browser compatibility or [cross-browser testing](https://www.geeksforgeeks.org/why-cross-browser-testing-important) is a process of evaluating whether the functionality of a web application remains consistent when the end-user visits your web application from a different browser.

It is also used to validate browser compatibility for desktop and web applications.

Some of the cross-browser testing tools are Sauce Labs, Browsershots, Browsera, Selenium Box, etc.

20. List some best practices in Test Automation.

Create an Automation Test Plan: Develop a plan outlining the strategy for identification and an approach for automation.

Identify test cases that can be automated: Identify the test cases that can be automated as not all test cases can be automated and should not be automated.

Test early and frequently: Starting testing early and running tests, again and again, will help to find more defects.

Create test data: Test data should be created carefully that exercises most parts of the application including boundary conditions and edge cases.

Select appropriate tools and frameworks: It is best to choose an automated testing tool that not only supports your application’s technology but should also be updated to support the latest technologies.

21. What is Test Automation Pyramid?

Testing Pyramid also known as Test Automation Pyramid describes the test types that the development and QA team should include in an automated test suite.

The purpose of the test automation pyramid is to provide rapid feedback to ensure that code changes do not impact the existing functionality.

It minimizes the time required for the developers to determine if an update was introduced.

22. What are the risks associated with Automated Testing?

Unrealistic expectation: Automation testing tools do not make a difficult test problem into an easy one so unrealistic expectation from the tool is the first and foremost risk.

Vendor issues: A free tool is made licensed over the course of time due to the inability to update the automation tools with changes in the software testing platform.

Maintainability of test scripts: In automation testing, it is important to carefully design maintainable, robust, and modular frameworks so that the test scripts and data could be reused and reduce the risk of lack of maintainability of scripts.

Underestimating the need for manual testing: As an automation test engineer it is important to understand that not all tests can be automated. With repetitive automated tests, there might be an issue called “Pesticide Paradox” where we do not find any new issues.

23. When will you not Automate Testing?

Tests that require user experience: Tests that require a user to respond as to how easy the app is to use cannot be automated.

Tests that will be executed one time: Tests that will be run only one time should not be automated as it will unnecessarily increase the project budget.

Tests that cannot be automated 100%: Tests that cannot be automated 100% should not be automated at all.

Tests that don’t focus on risk areas: Tests that don’t focus on risk areas of the application should not be automated at all.

24. How can you speed up an Auto Test Suite?

Reduce tests through AI: Tests through UI are generally slower so it is better to convert UI tests to API and unit tests wherever possible.

Parallelize your tests: Overall time needed to run the tests can be reduced by running tests in parallel.

Use Cloud-based testing solution: With the cloud-based testing solution tests can be executed on real devices in the cloud. A cloud-based approach can be faster and more reliable.

Invest in Continuous Integration: Continuous integration helps to reduce the time it takes to run your test suite by automating the build, testing, and deployment process.

Focus on critical functionality: It is a better option to focus on automating the most critical functionality instead of automating every single feature of your application.

25. Is automated testing making manual testing obsolete?

Automated testing is not making Manual testing obsolete. Although automated tests help to avoid regression issues there are some types of testing like usability testing, and accessibility testing that needs to be performed by a human tester.

Automation testing helps to test repeatable test cases which can be reproduced deterministically.

It helps to reduce the amount of manual testing required but it does not eliminate it.

26. What is UI Testing?

UI Testing is User Interface Testing a type of software testing that ensures the [UI](https://www.geeksforgeeks.org/user-interface-ui) of a website is user-friendly and meets the expected customer requirements.

It evaluates the design of elements such as layout, colors, fonts, font sizes, etc.

It allows testing of the functionality from the user’s perspective.

27. What is Protractor?

Protractor is an Automation testing framework that is written using [NodeJS](https://www.geeksforgeeks.org/nodejs) and offers combined end-to-end testing for web applications that are built using [AngularJS](https://www.geeksforgeeks.org/introduction-to-angularjs). It supports both Angular and Non-Angular applications.

The purpose is not only to test AngularJS applications but also to write automated regression tests for normal web applications.

It automatically executes the next step in tests the moment the webpage finishes the pending tasks.

28. Is Documentation necessary in Automation Testing?

Documentation plays a vital role in Test Automation. Having well-written documentation explains not only what, but also why it is important for the team members when they try to understand the tests. It is important to document all the procedures and methods, to ensure test repeatability.

29. What are the Pros of Selenium?

Open-source: Selenium is freeware and a portable tool that can be downloaded freely and community-based support is freely available.

Simultaneous execution of test cases: Selenium supports simultaneous execution of test cases in multiple environments and browsers.

Supports multiple operating systems: Selenium can operate and support multiple operating systems like Windows, Mac, [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system), and [Unix](https://www.geeksforgeeks.org/introduction-to-unix-system). The test suite can be created across any platform and then can be executed on another one.

Supports tools to diagnose failures: Selenium provides tools that are needed to diagnose the failures and rebuild a similar environment for new test execution.

Saves time: [Selenium](https://www.geeksforgeeks.org/selenium-basics-components-features-uses-and-limitations) Grid saves time as it uses a Hub-Node design.

Captures screenshots: Selenium captures the screenshot of browsers at different stages of test case execution.

30. What are the Limitations of Selenium?

No support for desktop applications: Selenium does not support testing for desktop applications.

Expertise: Selenium requires the expertise of your team — and resources to manage.

Maintenance and Scalability: Selenium is a maintenance-heavy framework — and is difficult to scale as one grows.

Open Source Forums: Since Selenium is open-source software, one has to rely on community forums to get your technical issues resolved.

No support for REST and SOAP Platforms: We can’t perform automation tests on web services like SOAP or REST using Selenium.

No Reporting capability: Selenium does not have any inbuilt reporting capability, one has to rely on plug-ins like JUnit and TestNG for test reports.

31. What are the features of a Good Automation Framework?

Ease of use: The testing framework should be easy to use, starting with the installation and ending with user-friendly test reports.

Reliability: The testing framework should be reliable and should be tested with the aim of preventing it from crashing.

Reporting: Reporting is one of the important features of the [testing framework](https://www.geeksforgeeks.org/8-useful-testing-tools-libraries-and-frameworks-for-react-developers). Test reports that are unreadable, and vague, make even the best test cases useless.

Integration: The testing framework should be easy to integrate with other tools.

Portability: The testing framework must be usable on [different OS](https://www.geeksforgeeks.org/different-operating-systems) and different environments.

32. Is Automation Testing a Black box testing or White box Testing technique?

Automation testing is mostly black-box testing as only the steps that a manual tester performs for software without knowledge of the code of the application are programmed.

33. What is Continuous Integration?

Continuous Integration is a practice in which the team of developers intermittently merges the code changes into the shared repository. In this developers are integrating multiple times throughout the day and committing to changes about once a day or a few times a week.

34. What is Continuous Delivery?

Continuous delivery lets automated testing developers to test beyond unit tests and helps them to verify the application updates across multiple dimensions before deploying to customers. With this every code change is built, tested, and then pushed to non-production testing.

35. What is the scripting standard followed while performing Automation Testing?

Uniformity in naming convention.

Adequate indentation.

Robust error handling.

Ability to recover easily.

Commenting on functionality whenever and wherever necessary.

36. Is it possible to conduct Automation testing without a framework?

Automation testing can be done without a testing framework but it is not recommended. The automation testing framework guides testers on how to execute the scripts, how to report the results, where to store them, etc. In short, the testing framework is like a blueprint.

37. Why do we need Cross-Browser testing?

Cross-browser testing helps to ensure that the application or the website delivers an optimal user experience independent of the browser that is used to access it. It helps to create a positive user experience on the website, no matter which browser or device the user uses to connect.

38. List the programming languages supported by Selenium Webdriver.

Selenium Webdriver supports programming languages:

[Java](https://www.geeksforgeeks.org/java)

[Python](https://www.geeksforgeeks.org/python-programming-language)

[Ruby](https://www.geeksforgeeks.org/ruby-programming-language)

[C#](https://www.geeksforgeeks.org/csharp-programming-language)

[JavaScript](https://www.geeksforgeeks.org/javascript)

[Perl](https://www.geeksforgeeks.org/perl-programming-language)

[PHP](https://www.geeksforgeeks.org/php-tutorial)

Automation Testing Interview Questions for Experienced

Here in this section, you will find all experienced-level questions for automated testing interview questions.

39. List the differences between Manual Testing and Automation Testing.

| Parameters | Manual Testing | Automated Testing |
| --- | --- | --- |
| Definition | In manual testing, the tester executes test cases manually. | In automated testing, the tester uses tools and scripts to automate the process of testing. |
| Working | Manual testing works by requiring analysts and QA engineers to be involved in everything right from test case creation to actual test execution. | Automated testing involves testers writing test scripts that automate test case execution. |
| Strength | Manual testing is better at handling complex scenarios. | Automated testing is much faster and covers many permutations. |
| Weakness | Manual testing is slow and tedious. | Automated testing requires coding and test maintenance. |
| Test coverage | It is difficult to ensure sufficient test coverage. | It is easy to ensure sufficient test coverage. |

40. List the main steps in the lifecycle of Automation Testing.

Decide the scope of test automation: This is the first stage of automation testing that involves deciding which modules can be mechanized and which not, which test cases can be automated, and how to automate them.

Choose the appropriate automation tool: Selecting the appropriate automation testing tool is a basic stage for an automation testing lifecycle. Some of the popular automation testing tools are Selenium, Appium, Cucumber, SoapUI, etc.

Plan, Design, and Strategy: During this phase, testers make an automation procedure and plan.

Setup the test environment: This phase involves setting up the test environment that involves setting up a machine where test scripts will be executed.

Test script execution: To execute the test scripts, the automation testing team must ensure that all the scripts are properly running. For this, they need to take notice of the processes like the test scripts should be created based on actual requirements, a common function method must be created to be used throughout the test process, etc.

Examine and Maintenance approach: It is the last phase of automation testing in which the testing team examines to determine whether the specific segments that experience an overall number of issue reports require an extra test attempt or not. The maintenance phase is used to validate whether the new functionality added to the product is working fine or not.

41. Some popular tools you would use for Automation Testing.

Selenium: Selenium is an open-source framework for web testing automation. It is a suite of software consisting of Selenium WebDriver, Selenium Grid, and Selenium IDE.

Katalon: Katalon is a scalable automation testing tool for web API, desktop, and mobile applications. It has flexible methods for test design recording and playback, manual, and scripting mode.

Appium: Appium is an open-source tool that is used for mobile applications. It allows users to write automated UI tests for native, web-based, and hybrid mobile applications on both Android and iOS.

TestComplete: TestComplete can automate functional UI testing for desktop, mobile, and web applications. It can handle and identify dynamic UI elements in most available technologies. It supports programming languages like JavaScript, Python, JScript, [C++](https://www.geeksforgeeks.org/c-plus-plus), etc.

Cypress: [Cypress](https://www.geeksforgeeks.org/difference-between-cypress-and-selenium) is an automation tool for end-to-end testing that can directly operate with the browser in the same run-loop as your application. It supports JavaScript programming language and integrations with popular CI/ CD tools.

42. List the criteria to select an automation tool.

Project Requirements: Automation tool selection should be based on the project requirements. The type of application that needs to be tested, platforms that need to be tested, and language application is built-in.

Team Skills: When selecting a tool for automation it is important to decide whether the project requires a codeless test automation tool or an automation tool that requires coding.

Budget: This is a very important criterion for selecting a testing tool. It is important to consider the cost of human resources, time spent on the learning tool, time being spent on automation, and the cost of infrastructure.

Reusability: It is important to look for tools that allow the reuse of the already created test steps to avoid writing the same code multiple times in multiple test cases and avoid duplication of efforts.

Support for Tools for Integration: Select a tool that integrates with [CI/ CD tools](https://www.geeksforgeeks.org/best-ci-cd-tools) that will be used in the project.

43. What is Modular Testing Framework?

The [modular testing framework](https://www.geeksforgeeks.org/module-testing) involves the tester dividing the application into modules and creating test scripts for each module. These test scripts are combined to build larger tests in a hierarchical fashion.

It involves less effort for creating test cases as test scripts for individual modules can be reused.

In case of changes to the application, only the module and its associated test scripts need to be fixed.

44. List the frameworks used in Automation Testing.

Below are some of the frameworks that are used in Automation testing:

Data-Driven Testing Framework: In a data-driven testing framework, testers can store and pass input/ output parameters to test scripts from an external data source.

Keyword-Driven Testing Framework: It is an extension of the data-driven testing framework that extracts test data from an external source and safely maintains the set of codes. These codes also known as keywords can be used to alter the test script to conclude possible outcomes from the test framework.

Hybrid Testing Framework: Hybrid testing framework combines the data-driven testing framework and keyword-driven testing framework allowing more scope for increased productivity and better success.

Modular Testing Framework: This framework requires the existence of various modules to run the framework and separate test scripts need to be created for every module and collaborate with them to create test cases for the application.

Linear Scripting Framework: Linear scripting is carried out manually and it takes up ample time to execute. It is also known as Record and Playback framework. It also requires time for maintenance.

Behavior-Driven Framework: Behavior-driven framework that allows the tester to create test cases in simple English. The test scenarios are easy to understand by everyone on the team.

45. What is Data-Driven Testing?

[Data-driven testing](https://www.geeksforgeeks.org/data-driven-testing-in-software-testing) is a software testing approach in which the test data is stored in the tables and it allows testers to input a single test script that can execute tests for all test data from a table and expect test output in the same table.

46. What is Keyword-Driven Testing Framework?

[Keyword-driven testing framework](https://www.geeksforgeeks.org/keyword-driven-testing-in-software-testing) divides test cases as test steps, objects of test steps, actions on test steps, and data for test objects.

It uses a table of keywords to represent the actions and inputs for each test case.

In this, the set of keywords is identified and then actions are associated related to these keywords.

47. What is Hybrid Testing Framework?

The hybrid testing framework is a combination of a data-driven testing framework and a keyword-driven testing framework.

It is mostly used by manual testers who don’t have knowledge of programming languages.

They can simply have a look at the keywords, test data, and object repository and start creating test cases without having to code anything in the framework.

48. List some features of the Cucumber framework.

The cucumber tool supports Behavior-driven development (BDD).

It offers to write tests in simple English that can be understood by anyone.

It uses Ruby programming language.

It focuses on end-user experience.

49. What is Robot Framework?

[Robot framework](https://www.geeksforgeeks.org/robot-framework-in-python) is an [open-source](https://www.geeksforgeeks.org/difference-between-free-software-and-open-source-software) test automation framework that is used for acceptance testing and acceptance-test-driven development. It follows different frameworks like data-driven, keyword-driven, and behaviour-driven development for writing test cases.

It provides support for external libraries and open-source tools that can be used for automation.

In this test cases are written using keyword style in a tabular format.

The most popular library used in the Robot framework is Selenium Library.

50. What are the different components of Selenium?

Different components of Selenium are:

Selenium Grid: Selenium Grid is used for parallel testing as it enables the execution of varied tests in multiple machines simultaneously.

Selenium IDE: Selenium IDE is available as Firefox and Chrome plugins. The recording, debugging, and editing of the functional tests can be done in Selenium IDE.

Selenium RC: It is the server implemented in Java that can accept commands for browsers using HTTP.

Selenium Webdriver: Selenium Webdriver receives commands and passes them to the browser. It supports browsers like Chrome, Firefox, Safari, IE, etc.

Why do we need Test automation frameworks?

Test automation frameworks are necessary for several reasons, including:

Efficiency: Automated tests can be run much faster and more frequently than manual tests, which can save a lot of time and resources.

Consistency: Automated tests can be run with the same inputs and expected outputs every time, ensuring consistent results.

Reduced human error: Automated tests remove the possibility of human error, which can lead to more accurate and reliable test results.

Reusability: Automated tests can be reused for different versions of the software, which can save time and effort.

Continuous integration: Automated tests can be integrated into continuous integration and continuous delivery pipelines, which can help identify and fix issues early in the development process.

Scalability: Automated tests can be used to test large and complex applications, which would be impractical or impossible to test manually.

You must ask 5 questions to the interviewer at the end of your interview.   
  
A. What is the team size and what is product road map  
B. How this position became available ( am I replacing someone or we are growing as a company so hiring more?)  
C. After researching the company website, I see great team culture, do you enjoy the same ?  
D. Will I be allowed to pair programs with dev and add automation scripts or will I get a chance to join diff sprints to work ?  
E. I covered most, is there any additional information you want to support my application… These are 5 questions if you ask , it will add 25% weight to your selection.

1) What major changes have you done in your framework, can you explain the same?

Ans: This is a very tricky question, you should share the exact changes you have made for your framework. For example, you can consider using Retry Listeners to rerun failed test cases. You also need to explain how you have implemented the same.

Refer to the link to check [How to Implement Retry Listeners in Automation Framework?](https://automationreinvented.blogspot.com/2020/08/how-to-rerun-only-failed-test-cases.html)

What is the Test Automation Framework?

Test Automation Framework is an Integral Part of Automation and it also makes sense also as designing & maintaining the Framework needs expertise. A test automation framework is a set of guidelines or rules that can be used to define test cases. These test cases can then be configured and implemented using test automation tools such as Selenium, Puppeteer, etc., to the delivery process via a CI\_CD Pipeline.

A test automation framework will consist of practices and tools that are designed to create efficient test cases. These practices range from coding standards, test-data handling methods, object repository management, and managing access control to test environment and external tools, etc.

2) How do you calculate ROI for Test Automation?

Ans: By calculating and demonstrating Test Automation’s Return on Investment (ROI), we can be better convinced that the investment will be worthwhile in the long run. Delivering more quantitative numbers on building and maintaining a test automation framework may be more helpful to get everyone on board for Automation.

To calculate Automation ROI first you need to first calculate the Manual Execution time of your test case. This value can be updated in your Test Management Tool.

You can use the formula to calculate ROI:

Time Saved by Automation  =Manual Execution Time -  Automation Execution Time

Automation ROI = Time Saved by Automation / Investment.

Investment = time required to build frameworks + maintenance cost + (time to code one tests X number of tests)

Hope you will be able to implement the same in your project.

3) Explain the challenges you faced while designing Framework?

Ans: This is a very tricky question to figure out how much exposure you have working with Automation. So sharing an honest answer would always be good.

Let me give an example: You are working on some service API testing and you are unable to perform the automation as you don't have access to some component, in that you can come up with an idea to create a mocking service integrated to your framework.

4) What is the reporting structure you use in your framework?

Ans: As we all know that proper HTML reporting is very important for Automation. So never answer that you don't have a reporting structure in your framework. There are many reporting ways available for framework, one of the most used ones is Extent Reporting.

You also need to provide a preview of the extent reporting implementation in Framework, please refer to the post here on [Extent Report Integration with Framework](https://automationreinvented.blogspot.com/2021/05/how-to-integrate-extent-reporting-to.html)

5) Have you worked on automation framework design from scratch?

Ans: This is a very important question and you should be completely honest with this answer, if you designed some framework then do explain what was the design patterns you have used and what all libraries you have integrated to design your framework.

Else you can say that you haven't designed anything from scratch but you have participated in Framework Enhancements in terms of CI-CD integration or HTML reporting.

6) Can you tell me why a Test Automation Framework is required?

1. We want to organize and standardize the test suite, which will further help in:

Reuse of test code

Ease with which test cases can be added

2. We want to be able to provide a way to organize groups of test cases. This will enable us to run specific tests.

3. Provide a layer of abstraction to protect the test suite against changes in service call protocols in order to minimize changes to the test code.

4. Decoupling the test cases from the automation code could lead to test automation misses, lack of visibility of mapping between the test case to test function. Hence, we want to provide mapping between test cases to its corresponding automation function.

5.  We want to be able to run our tests on any endpoint including the development desktops, all stages of the environment (DEVO/BETA/GAMMA etc), to enable integration testing on them.

6.  We want to be able to use standard tools for reporting test runs, collecting metrics, deployments.

7) How your team managing Automation Test Suites?

In our team we basically use Groups in TestNG to manage different Test Suites like Sanity, Regression.

TestNG allows us to perform sophisticated groupings of test methods. Not only can we declare that methods belong to groups, but we can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups while excluding another set. This gives us maximum flexibility in how we partition our tests. Example:

        @Test(groups={"sanitytest","regressiontest"})

8)  Do you know what is Mocking and have you ever used it in your framework or writing tests?

Ans: Mocking in programming refers to an action of substituting a part of the software with its fake counterpart. It is super useful with API Test Frameworks.

9) Can you tell me about the type of class you have in your utility package?

Ans: util package in Java is a package which contains various utility classes and interfaces. It provides basic functionality for commonly occurring use cases.

Some of the classes we have in our framework are:

 - ExtentReport

 - ScreenShotUtility

 - JsonReader

 - RetryListener

 - StringUtil

 - AssertionService

10) How are you scheduling your Automation Suite Execution?

The best way to manage your Automation Suite execution is by integrating the same with pipeline CICD, like Jenkins, CircleCI.

Check out the posts here to [Run Scheduled Automation Suite Job From Jenkins](https://automationreinvented.blogspot.com/2021/05/how-to-schedule-job-in-jenkins-pipeline.html)

11) If you want to execute your maven suite in dev-desktop or linux instance then what steps would you follow to install Maven into the Linux machine?

Ans: Check<https://automationreinvented.blogspot.com/2023/01/how-to-install-maven-on-linux-ubuntu-or.html> with all steps to perform Maven setup on Linux machine.

Explain Folder Structure in Test Automation Framework ?

<https://www.linkedin.com/posts/sidharth-shukla-77b53145_testing-automation-qualityassurance-activity-7098510719117119488-gJyQ?utm_source=share&utm_medium=member_desktop>

Lets answer a very important interview question for Automation QA and SDET Job Role: TELL ME ABOUT YOURSELF

🔴I'm a test automation engineer with 3-8 years of experience, depending on how you define it. I've always been passionate about technology, and I got my start in software testing during my initial phase of my career. Since then, I've focused my career on test automation, developing the skills and knowledge to create robust, reliable test automation frameworks.

🔴In my current role, I work closely with development teams to integrate test automation into the software development lifecycle.

🔴I have experience with a variety of tools and technologies, including Selenium, Appium, RestAssured, Maven, Git, Jenkins, and more.

🔴I'm also familiar with various programming languages, including Java, Python, and C#, which enables me to create customised solutions for a range of projects.

🔴Specifically, I have experience with Maven as a build automation tool, which allows me to manage project dependencies and streamline the build process.

🔴I'm also skilled in using Git as a version control system, which helps me to collaborate effectively with other team members and manage code changes efficiently.

🔴Additionally, I have experience with Jenkins as a continuous integration and continuous delivery (CI/CD) tool, which enables me to automate the testing and deployment process.

🔴I'm a detail-oriented problem solver, and I enjoy collaborating with others to find innovative solutions to complex challenges.

🔴Outside of work, I enjoy staying up-to-date on emerging technologies and industry trends, as well as spending time with my family and friends.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Check the link below for SDET specific Job Roles & Responsibilities:

<https://lnkd.in/gGuAfD48>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

How to perform pen actions using selenium ?

<https://automationreinvented.blogspot.com/2022/10/how-to-perform-pen-actions-using.html>

Scenario Based QA Interview Q&A

🔺 Scenario: Handling Flaky Tests

Question:

How would you deal with flaky tests in your Selenium automation suite using TestNG?

Answer:

To address flaky tests, I would implement retry logic in TestNG. By using the retryAnalyzer feature in TestNG, I can specify a custom retry analyzer class that determines whether a failed test should be retried based on certain conditions, such as specific exceptions or test result statuses. This helps improve the reliability of the test suite by rerunning failed tests automatically.

🔺 Scenario: Parallel Execution

Question:

Explain how you would implement parallel execution of tests in TestNG for faster execution in your Selenium automation framework.

Answer:

TestNG allows running tests in parallel, either at the suite or test level, by organizing them into different suites and configuring parallel attributes like "parallel" and "thread-count".

<suite name="MyTestSuite" parallel="tests" thread-count="5">

<!-- Test configurations -->

</suite>

🔺 Scenario: Data-Driven Testing

Question:

Describe how you would perform data-driven testing using TestNG in your Selenium automation framework.

Answer:

TestNG supports data-driven testing through its @DataProvider annotation, which allows me to supply test data from external sources such as Excel sheets or databases. I can create a method annotated with @DataProvider to provide test data, and then annotate my test methods with @Test(dataProvider) to execute the tests with different data sets. This enables to execute the same test logic with multiple input values and verify the expected behavior.

@DataProvider(name = "loginData")

public Object[][] getLoginData() {

return new Object[][] {

{"user1", "pwd1"},

{"user2", "pwd2"},

};

}

@Test(dataProvider = "loginData")

public void loginTest(String username, String password){}

🔺 Scenario: Grouping and Tagging Tests

Question:

How would you group and tag tests in TestNG for better organization and selective execution in your Selenium automation framework?

Answer:

TestNG allows grouping tests using the @Test(groups) annotation, enabling selective execution and better organization of test suites by defining groups in testng.xml. This allows for selective execution of tests and better organization of test suites like smoke, sanity, regression.

@Test(groups = {"smoke", "regression"})

public void loginTest() {

// code

}

Note:

Responses may differ, but I've provided examples for clarity. It's best to use examples specific to projects rather than general ones.

👉 Authentication Scenario:

Question:

You're testing an API that requires authentication using OAuth 2.0. How would you design test scenarios to ensure that the API correctly handles authentication tokens, including scenarios for obtaining, refreshing, and invalidating tokens?

Answer:

To ensure proper handling of authentication tokens in our API testing, I would design test scenarios covering the complete lifecycle of OAuth 2.0 tokens. This includes tests for obtaining tokens successfully, refreshing expired tokens, and verifying that invalid tokens result in appropriate authentication errors.

👉 Input Validation Scenario:

Question:

You're tasked with testing an API endpoint that accepts user input for creating a new resource. Describe how you would design test scenarios to validate various types of input data (e.g., strings, numbers, special characters) and ensure the API responds appropriately to invalid input.

Answer:

For testing the API endpoint that accepts user input, I'd craft test scenarios to cover a variety of input data types such as strings, numbers, and special characters. These scenarios will validate that the API responds correctly to valid input and appropriately handles invalid input, such as rejecting requests with invalid data types or exceeding length limits.

👉 Error Handling Scenario:

Question:

Consider a scenario where the API encounters an internal server error (HTTP status code 500) during the processing of a request. How would you verify that the API returns the correct error response with relevant error messages, and how would you handle and report such errors in your testing framework?

Answer:

In the event of an internal server error (HTTP status code 500), I would verify that the API responds with the expected error response containing relevant error messages. Additionally, I'd ensure that our testing framework is capable of detecting such errors, logging them appropriately, and generating reports for further analysis and debugging.

👉 Concurrency Scenario:

Question:

How would you approach testing the API's behaviour under high load or concurrent requests? Describe the strategies and tools you would use to simulate concurrent requests and analyse the API's performance, scalability, and response times under varying levels of load.

Answer:

To test the API's behaviour under high load or concurrent requests, I would employ strategies such as stress testing and load testing. Tools like Apache JMeter or Gatling can be used to simulate concurrent requests and analyse the API's performance, scalability, and response times under varying levels of load. These tests will help identify any bottlenecks or performance issues that need to be addressed.

👉 Data Consistency Scenario:

Question:

Imagine you're testing an API that interacts with a database to retrieve or manipulate data. How would you design test scenarios to ensure data consistency between the API responses and the underlying database state? Do consider scenarios involving CRUD operations, data integrity checks.

Answer:

When testing an API interacting with a database, I'd design test scenarios to ensure data consistency between API responses and the underlying database state. This includes testing CRUD operations to verify that data is correctly created, read, updated, and deleted through the API. Additionally, I'd perform data integrity checks to ensure that the data returned by the API matches the expected state of the database.

Behavioural & Situational Interview Q&A for QA

Behavioural Questions:

Can you describe a situation where you had to prioritise testing tasks to meet tight deadlines?

Answer: In my previous project, we had a critical release deadline approaching. I collaborated with the team to identify high-priority test cases and focused on testing the critical functionalities first. I also communicated with stakeholders to manage expectations and ensure that essential features were thoroughly tested within the given timeframe.

How do you handle disagreements with developers regarding reported bugs?

Answer: I believe in maintaining open communication and collaboration with developers. Whenever there's a disagreement about a reported bug, I provide detailed evidence and steps to reproduce the issue. I also encourage constructive discussions to understand the root cause and work together to find the best solution for the project.

Describe a time when you had to adapt to changes in project requirements midway through the testing phase.

Answer: In a recent project, we received updated requirements from the client during the testing phase. I quickly analysed the changes and adjusted our test cases accordingly. I collaborated with the team to ensure everyone was aligned with the new requirements and updated the test documentation to reflect the changes. This proactive approach helped us maintain the quality of deliverables despite the changes.

How do you ensure effective communication within your QA team and with other project stakeholders?

Answer: I believe in regular and transparent communication. I schedule daily stand-up meetings with the QA team to discuss progress, blockers, and any upcoming tasks. I also actively participate in sprint planning meetings and retrospectives to align priorities and address any issues. Additionally, I use tools like JIRA to track tasks and updates, ensuring that everyone is informed and on the same page.

Can you share an experience where you successfully implemented test automation to improve testing efficiency?

Answer: In a previous project, we identified repetitive test cases that were suitable for automation. I conducted a feasibility study and selected Selenium for web automation and Rest Assured for API testing. After developing the automation framework, we integrated it with Jenkins for continuous integration. This automation initiative significantly reduced manual effort, improved test coverage, and accelerated the release cycle.

Situational Questions:

You've encountered a critical bug in production after a recent release. Walk me through your approach to investigating and resolving the issue.

Answer: First, I would gather as much information as possible about the issue, including error logs, user reports, and any recent code changes. I would then reproduce the issue in a controlled environment and analyse the root cause. Once the cause is identified, I would collaborate with developers to develop and test a fix. After verifying the fix, I would coordinate with the release team to deploy the patch to production and communicate with stakeholders about the resolution.

Imagine you're assigned to a new project with complex business logic and limited documentation. How would you approach understanding the application and designing test cases?

Answer: I would start by conducting thorough exploratory testing to understand the application's functionality and behaviour. I would interact with developers, product owners, and other stakeholders to gather insights into the business logic and requirements. Based on my findings, I would create test scenarios and prioritise test cases to cover critical functionalities first. I would also document my testing approach and collaborate with the team to refine and validate the test cases.

You're tasked with implementing API tests for a new feature. How would you approach designing and executing these tests?

Answer: First, I would review the API documentation to understand the endpoints, request parameters, and expected responses. I would then design test cases covering various scenarios, including positive and negative test cases, boundary conditions, and error handling. Using tools like Rest Assured, I would automate these test cases and integrate them into the CI/CD pipeline using Jenkins. Finally, I would execute the tests against different environments and analyse the results to ensure the feature's reliability and stability.

Your team is adopting Agile methodologies for the first time. How would you contribute to ensuring a successful transition to Agile testing practices?

Answer: I would actively participate in Agile ceremonies such as sprint planning, daily stand-ups, and retrospectives to understand the team's goals and priorities. I would advocate for a collaborative approach to testing, emphasising early and continuous feedback. I would also encourage test automation and continuous integration to accelerate testing cycles and improve release quality. Additionally, I would share best practices and mentor team members to build their skills and confidence in Agile testing practices.

You've identified a regression bug in the latest release. How would you prevent similar issues from occurring in future releases?

Answer: To prevent similar regression issues, I would suggest implementing robust regression test suites covering critical functionalities and edge cases. I would also recommend enhancing our automated testing framework to include regression tests that run automatically as part of the CI/CD pipeline. Additionally, I would advocate for thorough code reviews and collaboration between developers and testers to catch issues early in the development

Describe a situation where you had to deal with a challenging stakeholder who had unrealistic expectations regarding the testing timeline. How did you manage the situation?

Answer: In a previous project, we had a stakeholder who demanded an unrealistic testing timeline due to aggressive project deadlines. I initiated a meeting with the stakeholder to discuss the constraints and risks associated with the proposed timeline. I provided insights into the testing process, explained the need for thorough testing to maintain quality, and presented a revised timeline that balanced project deadlines with testing requirements. By effectively communicating the trade-offs and collaborating on a realistic plan, we reached a consensus that satisfied both the stakeholder and the testing team.

Imagine you're assigned to a project with a limited budget for testing resources. How would you prioritize testing efforts to maximize coverage within the budget constraints?

Answer: I would start by conducting a risk assessment to identify critical functionalities and areas of the application that require thorough testing. Based on the assessment, I would prioritize testing efforts on high-risk areas that could have a significant impact on the project's success or end-user experience. I would also leverage automation tools like Selenium and Rest Assured to automate repetitive test cases and optimize testing efficiency. Additionally, I would explore crowd testing or collaboration with other teams to supplement testing efforts within the budget constraints.

You're part of a cross-functional Agile team, and a new feature requires coordination between multiple teams for testing. How would you ensure effective collaboration and alignment between teams to deliver high-quality results?

Answer: I would start by establishing clear communication channels and regular sync-ups with all relevant teams involved in testing the feature. I would encourage cross-functional collaboration by organizing joint testing sessions, where testers, developers, and product owners can collaborate to validate the feature's functionality and integration. Additionally, I would leverage Agile ceremonies like sprint planning and retrospectives to address any bottlenecks or dependencies and ensure alignment on testing priorities and goals across teams.

You've been tasked with improving test coverage for a legacy application with minimal existing test documentation. How would you approach this challenge?

Answer: I would start by conducting a comprehensive analysis of the application's functionalities and identifying areas with the highest risk and impact. I would then prioritize test coverage efforts based on the criticality of these functionalities. Using techniques like exploratory testing and risk-based testing, I would create test scenarios to cover critical paths and edge cases. I would also work closely with developers to understand the application's architecture and identify areas where automated tests can be implemented using tools like Selenium and Rest Assured.

Describe a situation where you had to troubleshoot a complex test automation failure. How did you identify the root cause and resolve the issue?

Answer: In a recent project, we encountered a recurring test automation failure that was impacting our release cycle. I started by analyzing the test logs and error messages to identify patterns and potential causes of failure. I then reviewed the test scripts and environment configurations to pinpoint any discrepancies or dependencies. After identifying the root cause, I collaborated with the development team to implement a fix and re-run the automated tests to validate the resolution. I also updated the test framework and error handling mechanisms to prevent similar failures in the future.

Scenario Based QA Manager Interview Q&A

Leadership and Team Management:

Scenario: Your team is falling behind on a project deadline. How would you handle the situation?

Question: How do you motivate and manage your team to meet tight deadlines?

Answer: I would conduct a quick team meeting to identify challenges and bottlenecks. Then, I'll delegate tasks based on team members' strengths, monitor progress closely, and provide necessary support. Regular check-ins and positive reinforcement will keep the team motivated.

Quality Assurance Process:

Scenario: A new feature is being introduced, and you need to ensure thorough testing. How would you design the testing process?

Question: Can you outline your approach to implementing a comprehensive testing process for a new feature release?

Answer: I'd start with a detailed test plan, covering functional, performance, and security testing. Collaborating with development, I'll define test cases, prioritise testing areas, and establish a feedback loop for continuous improvement.

Handling Bugs and Defects:

Scenario: A critical bug is discovered in a production environment. How would you respond?

Question: How do you prioritise and handle critical bugs that are affecting the live system?

Answer: I'd immediately assemble a cross-functional team to assess the impact and identify a quick fix. Communication is key—keeping stakeholders informed about the situation and providing an estimated resolution time.

Test Automation Implementation:

Scenario: The organisation wants to increase test automation coverage. How would you approach this initiative?

Question: What steps would you take to implement test automation in a team that has limited experience with automation?

Answer: I'd start with a feasibility study, assess skill gaps, and provide training where needed. Gradually, I'll introduce automation frameworks and mentor the team to write and maintain automated tests.

Conflict Resolution:

Scenario: Two team members are having a disagreement that is affecting team dynamics. How would you resolve this conflict?

Question: Describe a situation where you successfully resolved a conflict within your team.

Answer: I'd organize a private meeting with both individuals to understand their perspectives. Encouraging open communication and finding a common ground would be my approach. If needed, I'll involve HR for further assistance.

Vendor Management:

Scenario: Your team is considering outsourcing some testing activities. How would you evaluate and manage a testing vendor?

Question: How do you assess and select a testing vendor for external testing services?

Answer: I'd start with a detailed vendor evaluation, considering their expertise, track record, and cost-effectiveness. A clear SLA and regular performance reviews would be established to ensure quality.

Continuous Improvement:

Scenario: Your team has completed a project. How do you conduct a retrospective to improve processes for the next project?

Question: Describe your approach to continuous improvement in a QA team.

Answer: I'd conduct a retrospective meeting with the team, analyzing what went well, areas for improvement, and lessons learned. Actionable items would be identified and implemented in subsequent projects.

Stakeholder Communication:

Scenario: You need to communicate testing progress and results to non-technical stakeholders. How do you approach this?

Question: Explain how you tailor your communication about testing progress to non-technical stakeholders.

Answer: I'd create concise and visually appealing reports highlighting key metrics and testing outcomes. Holding regular meetings or sending status updates would keep stakeholders informed and engaged.

Adapting to New Technologies:

Scenario: The company is adopting a new technology stack. How would you ensure your QA team is prepared for this change?

Question: How do you keep your QA team updated and skilled in new technologies?

Answer: I'd organize training sessions, encourage self-learning, and possibly bring in external experts for workshops. The team would also be involved in pilot projects to gain hands-on experience.

Metrics and KPIs:

Scenario: Management wants to see metrics indicating the effectiveness of the QA process. What metrics would you choose, and how would you present them?

Question: Can you outline the key performance indicators (KPIs) you would use to measure the success of your QA team?

Answer: I'd focus on metrics like test coverage, defect density, and release quality. Clear, visual dashboards would be created for easy interpretation by both the team and management.

Coforge final interview 2024:

1. Techniques of Blackbox testing.

2. ⁠Example for Boundary value Analysis.

3. ⁠Exception thrown when xpath is not locating when findElement is used.

4. ⁠Diff between findElement and findElements

5. ⁠StaleElementReference exception

6. ⁠Method overloading and method overriding example in framework.

7. ⁠Can constructor  be overridden?

8. ⁠Can value of static variable be changed? If yes, how?

9. ⁠What is dry run in Cucumber?

10. ⁠Difference between Scenario and Scenario Outline?

11. ⁠Before method and before test in testng?

12. ⁠Factory annotation in testng?

13. ⁠git clone and git fork?

14. ⁠git pull and git fetch?

15. ⁠Types of integration testing

16. ⁠Postman code to fetch value of particular node

17. ⁠302 status code?

18. ⁠How to pass username and password in header of api request without exposing in rest assured?

19. ⁠Program to find duplicate characters in a given string with its occurance(Should not use hashmap)

|  |  |
| --- | --- |
| JAVA CODING INTERVIEW Q&A BANK: 240 Coding Q&A | <https://drive.google.com/drive/folders/1x8CFUXmC7_3UZlsPuqa3QO2JwScM-YFL?usp=sharing> |
| Top 50 JAVA Interview Q&A: | <https://drive.google.com/file/d/17gBYGa8WvKnsR4bUJsq8vD_eeotbO7o5/view?usp=sharing> |
| Workshop on transitioning from Automation QA to SDET with a comprehensive learning plan from basic to advanced levels: | <https://drive.google.com/file/d/1g6lXRMhCci5fudfRqyiGyiGqIr1CbhjA/view?usp=sharing> |
| TEST PLAN TEMPLATE: | <https://docs.google.com/document/d/1LPxIe6QpnTTIapw-L_7hkJUl7kl0KBVK/edit?usp=sharing&ouid=104128041009417677699&rtpof=true&sd=true> |

PS Interview Questions:

Interview question for nearly 4-6YOE

First Round: Technical 30min

1. Intro about yourself

2. ⁠Fluent wait syntax

3. ⁠Explicit and implicit wait

4. ⁠findElements and findElement

5. ⁠programs on String Manipulation, Hashmap

6. ⁠difference between list, set, map and usage in framework

7. ⁠Api response codes

8. ⁠RestAssured post request syntax example

9. ⁠difference between selenium 3.0 and 4.0 version

10. ⁠basic sql queries

11. ⁠cross browser testing

12. ⁠Extent reports

13. ⁠Cucumber - how to run test cases parallely without testng being integrated

Second Round: Technical 1hr 40 min

1. Introduction about yourself

2. ⁠difference between static and instance

3. ⁠oops concepts and usage in current framework

4. ⁠ what are the reusable methods or implementation done in framework by me

5. For⁠ Coding - they will provide a n online compiler link separately for executing the programs

Program on how to create treemap and print the values

Program on a given input string to print the letters, digits, special char separately as output

6. Writing Xpaths in flipkart website

7. ⁠what is dynamic webtable and how to write xpaths for it

8. ⁠questions on waits

9. ⁠what is garbage collector and how it is invoked in the script

10. ⁠use of cucumberoptions

11. ⁠cucumber framework explanation

12. Selenium Code for ⁠switching to required window and close it then come back to the default window

13. ⁠explain exceptions faced during execution

14. ⁠what is Apache POI

15. ⁠try, catch, finally

16. ⁠mouse hover actions

17. ⁠git commands

18. ⁠STLC

19. ⁠jenkins integration and setting up ci/cd pipeline

20. Testing strategies ⁠

21. What ⁠defect clustering, pesticide paradox, testing pyramid

22. ⁠agile methodology and ceremonies

23. ⁠bug lifecycle

24. ⁠maven commands

25. ⁠RestAssured API Assert syntax

26. ⁠regression testing

Third: Managerial Round 40 min

1. Self intro

2. ⁠what makes one a good Quality tester

3. ⁠risk based testing, performance testing

4. ⁠scenario and behavioural based questions

5. ⁠what are the reasons that can cause a test case to fail

6. ⁠when jobs are scheduled in jenkins and next day it fails due to unkown reasons. What would one do to self heal automatically

7. ⁠what are the ways to complete exexution where there are more numb of cases and less time

8. ⁠negative scenario for an e-commerce domain

9. ⁠application architecture of current proj ( front end, back end, third party tools in-between)

10. ⁠what are non func testing explain