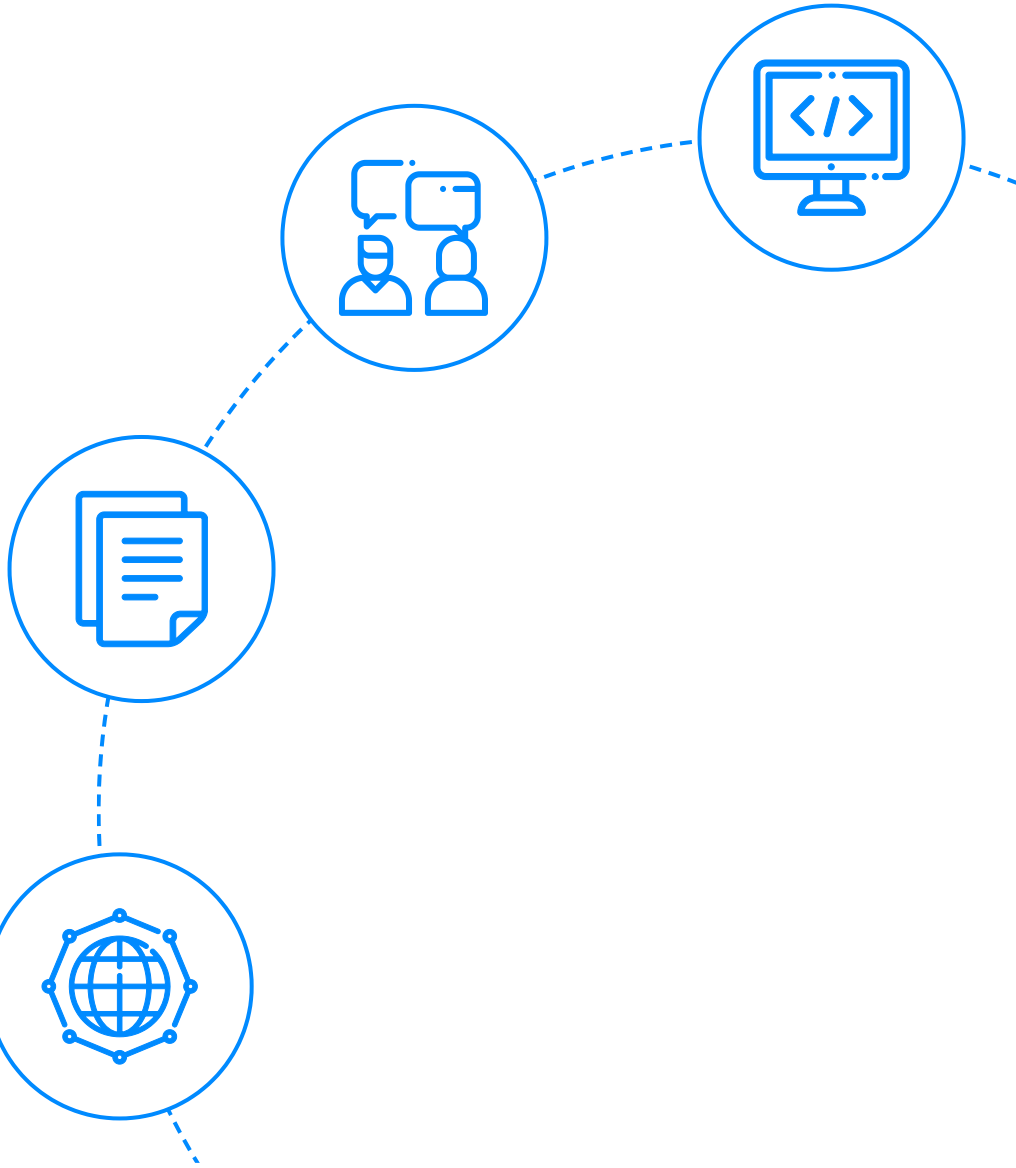
InterviewBit

Automation Testing Interview Questions



To view the live version of the page, click here.

Contents

Automation Testing Interview Questions For Freshers

Automation Testing Interview Questions For Experienced

Automation Testing Interview Questions For Experienced

(.....Continued)

19. What are the different components of Selenium?
20. What is UI testing?
21. What is Protractor?
22. What is a test automation platform?
23. What are some of the alternatives to Selenium?
24. What is the Robot framework? Provide a brief overview of its architecture.
25. What are the test library APIs provided by the Robot Framework?
26. How will you automate the basic login in a web application?
27. What are some risks associated with automated testing?
28. What are the different phases in an automation testing life cycle?
29. What is CAPTCHA?
30. How do you automate the testing of CAPTCHA?
31. What are some development practices to follow when writing automated tests?
32. When selecting an automation tool, what features will you look for?

Conclusion

33. Conclusion

Let's get Started

We all make mistakes. Software development is not an exception. No matter how careful you are when writing code, there is a chance of introducing bugs in the system.

However, this cannot be used as an excuse when delivering software to customers. There should be some process in place between the development and release to ensure high quality in the software. Software testing is the process of finding bugs and errors in the software. Its goal is to give enough confidence to the stakeholders to release the software to clients.

In general, you can divide software testing into two broad categories, namely manual and automated testing.

- **Manual Testing:** A human tester manually tests the software, performing all the actions that the real users are supposed to take.
- **Automated Testing:** A software tool conducts the testing, programmatically executing the code under the test, providing the pre-configured inputs, and verifying the actual output with the expected result.

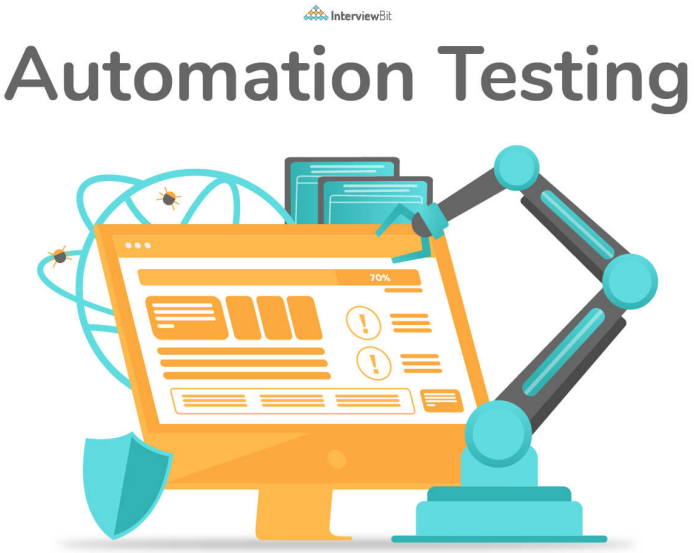
Both manual and automated testing have their advantages and disadvantages. A healthy software development organization uses both of these techniques to ship high-quality software.

This article provides the important questions that an interviewer may ask for a software tester position. Its primary focus is on automated testing. We have divided the questions into two categories, for beginners and for advanced.

Automation Testing Interview Questions For Freshers

1. What is automation testing?

Automation testing is a software testing strategy in which a tester programmatically runs the tests using a tool or a framework instead of manually going through the test cases and executing them one by one.

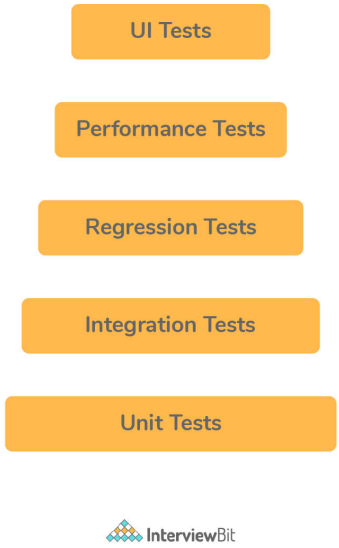


The primary goal of automated testing is to save time, effort, and money on repetitive tests that don't change frequently.

Automation testing helps teams and organizations automate the testing efforts, in turn reducing the need for human intervention and thus achieving greater speed, reliability, and efficiency. It also helps speed up the development cycle, as the developers get quick feedback and can iterate quickly.

2. What are the types of automation testing?

There are different testing techniques, but you can not automate them all. For example, exploratory testing. Here are some testing techniques that you can automate.



1. Unit tests: These are written by software developers and test a unit of code in isolation.
2. Integration tests: These test how well different software components work with each other.
3. Regression tests: Verify that the new code didn't break any existing functionality.
4. Performance tests: Ensure that the software won't crash and perform reasonably under heavy load or stringent conditions.
5. UI tests: Ensure that the software uses a consistent user experience and no visual or graphical elements on the screen are broken.

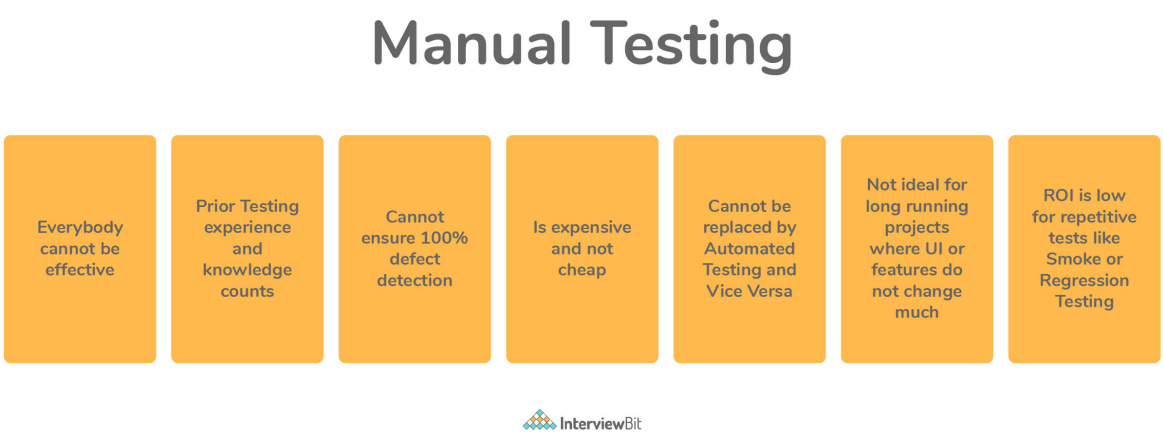
3. What's the difference between manual testing and automated testing?

Manual Testing	Automated Testing
A human executes the test cases one by one, without any software assistance.	Tests are executed by a testing tool or framework, without human assistance.
Useful for non-repeatable tests that involve human ingenuity, participation, and domain experience.	Useful for repeatable tests where the software feature under test doesn't change frequently.
Good for accessibility and usability testing, as the tester can test the software from an end-user's perspective.	Good for testing regression issues to make sure that the software didn't break after introducing new changes.
Can be slow and time-consuming, and subject to human errors and misjudgment.	Since it's run by a computer, automated tests are fast and free from errors, given that we are testing the right thing.
It's possible to test the software in a randomized manner, also known as exploratory testing.	Exploratory testing is not possible in automated testing.
UI problems and inconsistencies are easily spotted by a human tester.	Unless it's programmed for that, the automated testing cannot discover and report the UI problems.
It's very difficult, rather impossible to test the software under extreme load to conduct performance testing	Performance testing can be easily done with automation testing.

4. When is a good time to automate a test?

A test is a good candidate for automation under the following conditions.

- The test is repeatable.
- The feature under the test doesn't change its behavior frequently.
- It's time-consuming for a human tester.
- The test involves complicated calculations.
- The test ensures the previous functionality didn't break after a new change.



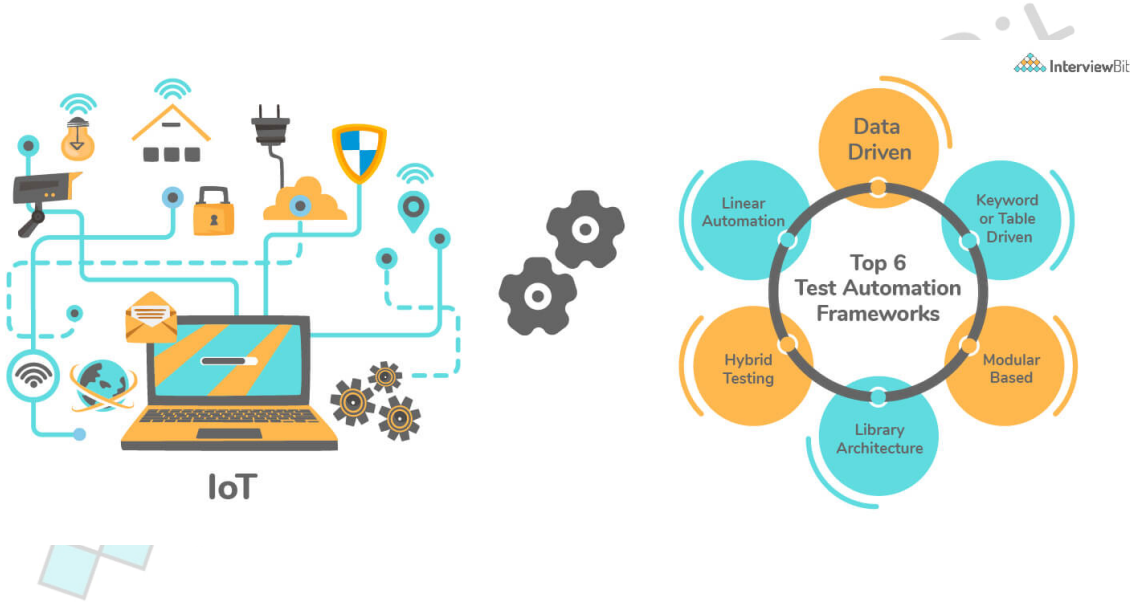
5. When will you avoid automated testing?

Though automation has its advantages, it's not a good idea to automate all of your testings. Here are some scenarios when a human tester can do a much better job of testing the software than an automated test suite.

1. The software or the feature under the test changes frequently. It means you have to update your automated tests often to keep them up to date. Tests can quickly become obsolete and stop providing any value.
2. Automated testing is also not suitable for exploratory testing. A human tester can explore the software in a much better way than a computer.
3. Unless the automated tests are programmed or configured to look for UI issues, they can't find any problems with the UI. It's much efficient for a human tester to spot any UI inconsistencies or design issues.

6. How do you choose a tool/framework for automated testing?

To perform any automation testing, you need to rely on software tools or frameworks. There are plenty of options to choose from many alternatives.



Here are some criteria based on which one can evaluate these tools.

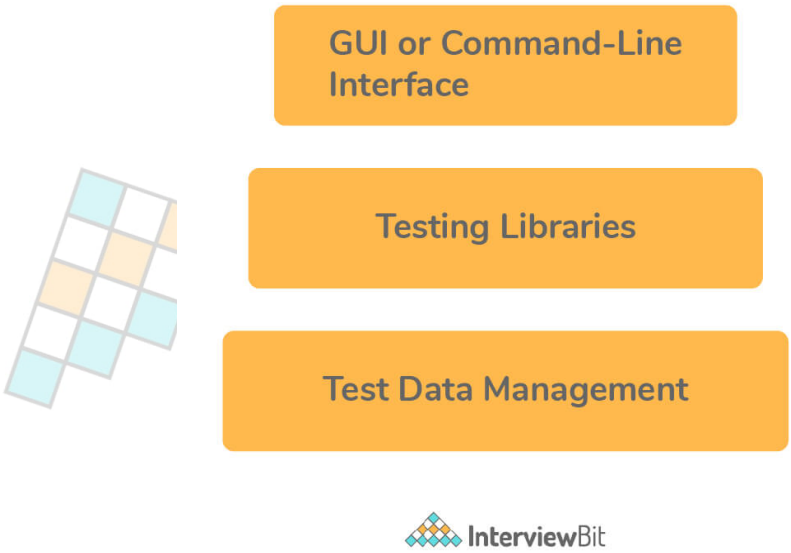
- **Programmable (code-based) or code-less tools.** Some tools require programming skills, while some don't, allowing a non-coder tester to create test cases with visual assistance. Depending on your team's experience and skill-set, you should choose accordingly.
- **Commercial vs. Open Source.** There's a vast variety in the pricing of the tools based on the feature they have. Commercial tools can be expensive, but you get tech support when needed. Open-source software is free, but you have to do your research when troubleshooting the problems.
- **Ease of use.** Some automated testing tools are notoriously hard to use and require extensive training before providing any value. Some are easy to use, and you can start using them out-of-box.

Some of the most popular automation tools include Selenium, Katalon Studio, UFT, TestComplete, Testim, etc., and many more. When choosing one, you should consider the testing requirements for your project, consult your team, and assess their skills, experience, and comfort with the tool.

You should also periodically assess the return on investment from the tool you choose and be prepared to switch if needed.

7. What are the different parts of a test automation framework?

A test automation framework makes it easy to perform automation testing for your software. Here are some components of a test automation framework.



- **Test Data Management**
  - A big problem in automation testing is generating the test data. A good test automation framework makes it easy to build test data for the application under test.
- **Testing Libraries**
  - Managing and running the automated tests is a crucial component of any automated testing strategy. A test automation framework provides libraries that make test management easy.
  - A good test automation framework provides support for unit tests, integration tests, and end-to-end tests.
- **Testing Tools**
  - Includes any GUI or command-line tools that make it convenient for testers to run the set of tests repeatedly.
  - It also consists of tools that enable testing the software under high load to conduct performance testing.

8. Should you automate all testing?

Although test automation has its advantages, it is not practical to automate all kinds of testing. Some testing can be done only by a human tester, such as user interface testing, usability, and accessibility testing.

Exploratory testing is another type of testing where a human tester provides more value than an automated test. In exploratory testing, a tester explores the software randomly, just as an end-user would do, and tries to find the bugs or UI inconsistencies or any hidden problems that developers might have overlooked.

Automated testing is helpful for large projects involving complicated calculations and for repeatable test cases. For features that change often and rarely executed test cases, a human tester provides a bigger ROI than automation would.

9. What is a test environment?

A test environment is a computer or a server on which a tester tests the software. After the team builds the software, the tester installs it on this computer with all its dependencies, just like the production environment. This allows the tester to test the software in a real-world scenario.

A test environment enables the tester to create reliable test setups which are identical whenever a new version of the software is released. The test environment includes the test bed, which is the test data using which the tester will test the software. This data helps the tester to verify test cases that need a particular setup.

Typically, the test environment is an identical copy of the production environment. Having a duplicate copy allows the tester to reliably reproduce the bugs reported by the customers and provide the exact steps to the developers to fix them.

Here are some prerequisites for a good test environment:

1. A server with a similar configuration, including the software and the hardware to match a production environment.
2. Sample test data with which to test the software.
3. Test database with reasonably realistic data, it can be a copy of an actual production database.
4. Installed software under the test.

10. What is browser automation?

Browser automation is the technique of programmatically launching a web application in a browser and automatically executing various actions, just as a regular user would. Browser testing gives you the speed and efficiency that would be impossible for a human tester. Protractor, Cypress, and Selenium are some of the popular tools used in-browser testing.

Some of the activities performed in browser automation are as follows:

- Navigate to the application URL and make sure it launches
- Test the various links on the web page and ensure they are not broken.
- Keep a record of the broken links on the page.
- Perform load and performance testing on your web application.
- Launch multiple instances of the browsers with different test users and ensure that concurrent actions work as expected.

11. What is cross-browser testing?

With web applications, you don't know in advance which browsers your users will use. Hence, it's crucial to test the web application or the website on multiple major browsers running on different operating systems.

Cross-browser testing is a type of browser automation testing where the tester verifies if the web application will work smoothly on different browsers. Some of the popular browsers include Google Chrome, Mozilla Firefox, Internet Explorer, Safari, etc.

The goal of the cross-browser testing is to launch the application on various browsers running on different operating systems, e.g. Windows, Mac OS, Linux, etc., and verify that the application works as expected. The tester looks for the design/rendering issues, the functionality of the application, and device-specific functionality.

Though it can be typically, sophisticated tools exist that allow the testers to automate cross-browser testing. Some examples include Selenium Box, BrowserStack, Browsershots, LambdaTest, etc.

12. Why do you need cross-browser testing?

With web applications, you can't guarantee the browsers/platforms/devices your users might use to access your software. Some users could be using Google Chrome on their Android phones, some might use Firefox on a Windows desktop machine, or others could use Safari on their Macbooks.

Cross-browser testing ensures that your web application works as expected on different versions of popular browsers on multiple platforms and devices. It ensures that the users get the same experience and features irrespective of which browser they use. It helps to reach a wide range of users, allows the users to switch browsers and devices, and still get the same user experience, increasing customer satisfaction and building a loyal user base.

13. What is automated regression testing?

Software is never done. The developers are constantly adding new features, functions, fixing bugs, and so on. There is a chance that all this new code might break the existing functionality that was working.

Users dislike using a product that is broken after they download and install a new release. They expect a consistent and reliable experience from the software, no matter which version they are using. They also expect that previously working features will keep on working and won't break in the future.

Regression testing is a testing technique where a tester makes sure that the new features didn't break any existing functionality. Its goal is to ensure that previously developed and tested functionality still works after adding new code. When a tester performs the regression testing automatically using testing frameworks and tools, it's known as automated regression testing.

In automated regression testing, a tester runs the suite of regression tests after each new release of the software. If the tests pass, then the tester continues with other types of testing. However, if it fails, then there is no point in further proceeding with tests until the developers fix the broken regression tests. Hence, they also act as a time-saver for the tester and ensure quality in software before shipping it.

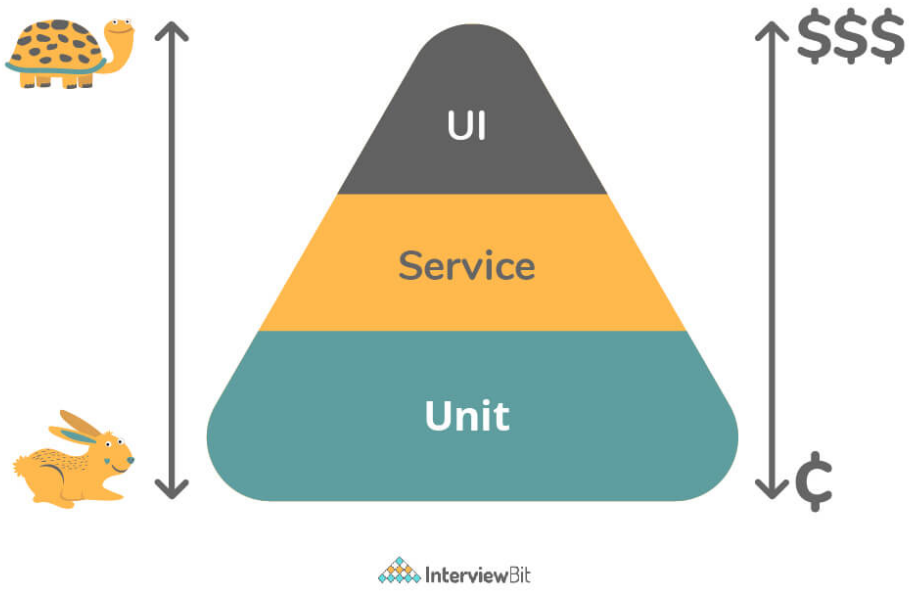
14. What are some of the best practices in test automation?

Here are some of the best practices a software development and the testing team should use to ensure quality software.

- **Decide what to automate**
  - It's not possible or practical to automate certain tests, such as usability, accessibility, exploratory testing, or non-repetitive test cases that frequently change.
- **Assign test cases based on skill and experience**
  - When dividing test cases, take into account the skills and experience of the tester and the complexity and severity of the feature under test.
- **Removing Uncertainty**
  - The whole goal of test automation is to have reliable, accurate, consistent tests that provide helpful feedback to the tester. If the tests fail due to bugs in the test itself, or it's giving false positives, then the ROI on test automation starts decreasing.
- **Choosing the right frameworks and tools**
  - There are a lot of tools to perform automation testing. Picking the wrong tool for the test at hand will waste time and provide false confidence to release software that may fail in production.
- **Keeping test records in a bug database**
  - Using a bug database is a best practice whether a team uses test automation or not.
  - Whenever new bugs are found by the automation tool or by the testers, they should be recorded in a bug tracking tool with the exact steps to reproduce the bugs and other details.

15. What is the test automation pyramid?

Martin Fowler first proposed the concept of the test automation pyramid[1] in 2012. It's a technique to think about how you should use different types of test automation to get the maximum value out of them.
The main idea behind the test pyramid is to have many unit tests and a few broad tests for the GUI.



GUI testing is very brittle. User interfaces are constantly changing. An enhancement to the software easily breaks up many tests, which need to be updated, causing additional work for the team. Testing the UI is slow and results in increased build times. You can perform it on a few machines on which you have the license for the GUI testing tool.

Hence, the test pyramid argues that you should have more automated unit tests than through the traditional UI-automation tests. It also has an intermediate layer of service tests that can provide many benefits of end-to-end UI tests without the complexities of dealing with the UI frameworks.

Automation Testing Interview Questions For Experienced

16. Is automated testing making manual testing obsolete?

No. Automated testing is not making manual testing obsolete. Though automated tests help avoid regression issues or find problems that you are already aware of, manual exploratory testing is essential to find the bugs you don't know about, such as incorrect requirements or implementations.

Some types of testing, such as exploratory testing, usability, and accessibility testing, need to be performed by a human tester.
Automated testing is only as good as automated tests. If bugs or problems are in the tests themselves, they will provide wrong results, giving false assurance to the stakeholders.

Good automation testing tests repeatable test cases which you can reproduce deterministically. It certainly reduces the amount of manual testing that a human tester would perform but does not eliminate it. Once a human tester discovers a bug, they can add automation tests to ensure that it's caught automatically in the future.

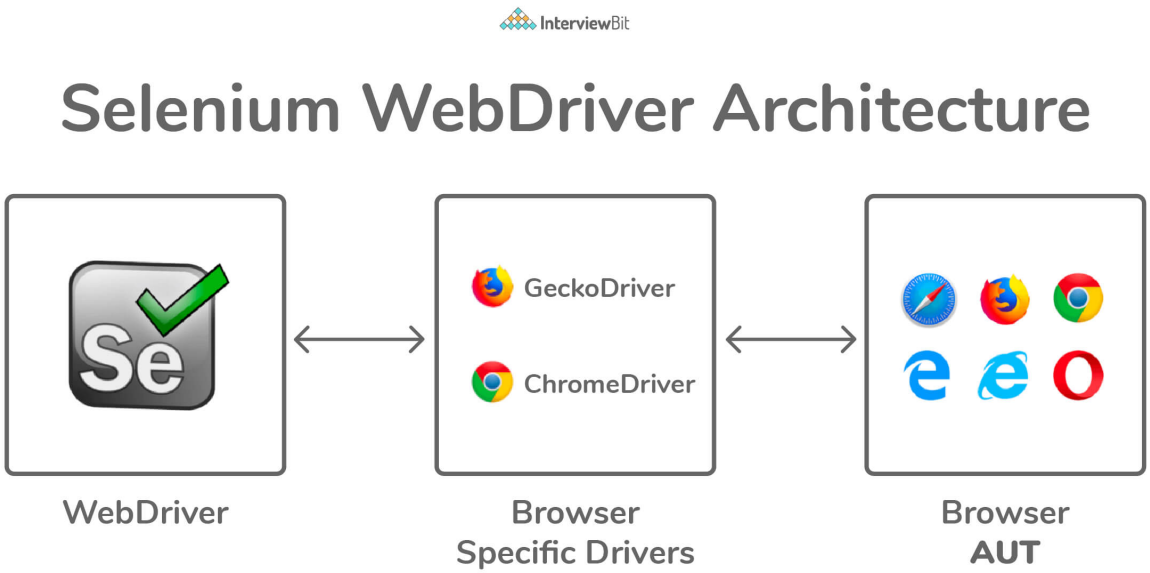
17. Who should be responsible for test automation? Developers or the QA?

As a team is supposed to be a single unit responsible for shipping a quality software system, it's a team's responsibility to write, execute, and manage test scripts. That means the developers and the QA should collaborate and use each others' skills to perform automation testing effectively.

Each team has a different skill set. Some have more technical testers, some have quality-focused developers, and some have testers who came from development backgrounds and vice versa. Hence it's not a good idea to pigeonhole automated testing to a particular department. Instead, collaboration and working together is the key to a successful automated testing strategy.

18. What is Selenium? What are its pros and cons?

For any web application, browser automation and cross-browser testing are two critical testing activities to ensure that the software works on various browsers/devices/platforms. Selenium[2] is a popular web automation tool that helps achieve that. It's one of the most widely used and popular tools used in automation testing.



Advantages of Selenium:

- Open Source: It's developed in open and has excellent community support. The software is updated regularly, ensuring significant problems and bugs are fixed, and new features are getting added constantly.
- Cross-Browser: Selenium allows you to run and test your web application in multiple browsers, such as Chrome, Safari, Firefox, etc.
- Cross-platform: You can use Selenium on Windows, Mac OS, or Linus. This allows testing the platform compatibility of your web application.
- Language Agnostic: You can use Selenium in your favorite programming languages, such as Java, C#, Python, Ruby, and many more.

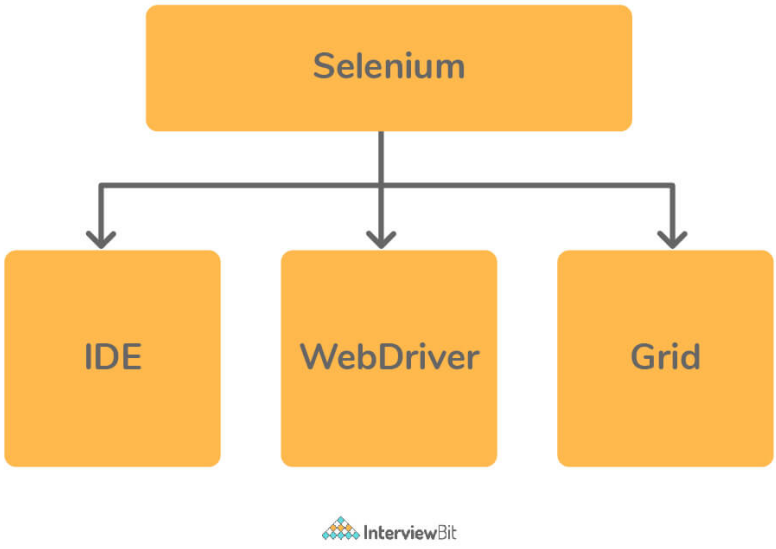
Disadvantages of Selenium:

- Learning Curve: One of the most common and recurring problems mentioned by new testers is that Selenium is complicated and takes a long time to learn. It requires prior programming knowledge.
- No support for desktop/mobile: Selenium only supports web applications. You cannot use it to test your desktop and mobile applications.
- No reliable tech support: As it's open-source software, there's no dedicated tech support that you can use in case you run into problems.
- Complicated debugging: It's tougher to debug Selenium programs than the other tools and frameworks.

19. What are the different components of Selenium?

Selenium is not a single tool or a framework. It is a suite of tools that work with each other or in isolation to provide different types of automation testing. Here are the four major components of Selenium.

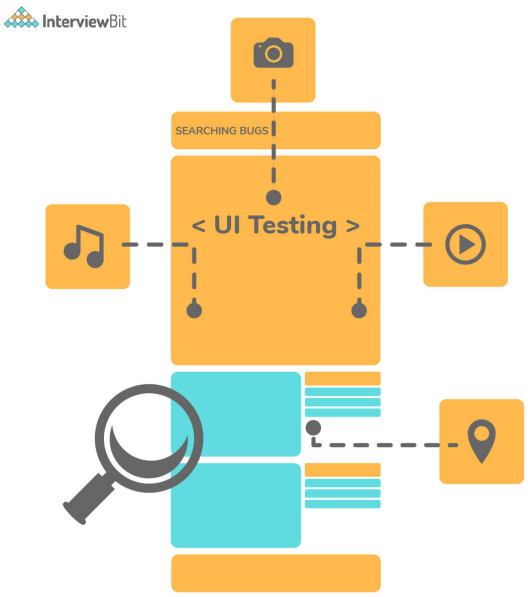
- **Selenium WebDriver**
  - A collection of open-source APIs and browser-controlling code implementations that provide a concise and straightforward programming interface.
- **Selenium Grid**
  - It enables the tester to run multiple tests across different browsers, machines, and operating systems in parallel.
- **Selenium IDE**
  - Stands for the Integrated Development Environment.
  - It allows the tester to write, record, run and debug the test cases.



20. What is UI testing?

The software's user interface is the only thing that the users see, touch and feel. They are not concerned about the backend code, database, or the frameworks you used to build the software. Building an application with broken, inconsistent, or annoying user interfaces can be enough to lose customers.

The goal of the UI testing is to ensure that the software uses a consistent user experience and no visual or graphical elements on the screen are broken. An advanced form of UI testing also ensures that the user interface is intuitive, prevents common mistakes, and doesn't get in the way of the users getting their job done.



Typically, UI testing is performed manually by a human tester. With the advancements in the tools and frameworks in automation testing, UI testing is becoming a good candidate for automation.

21. What is Protractor?

Protractor is an open-source automated testing framework that allows you to perform end-to-end testing of your web applications. It's built on top of WebDriverJS. Protractor is developed by Google and is especially used for testing Angular applications.

Protractor runs the tests against the web application by running it in real web browsers. It also interacts with the application like an end-user would, e.g. clicking buttons, links, filling forms, etc., and verifying the result with the expected outcome.

Since Protractor is based on the Selenium WebDriver, it's easy to perform cross-browser testing. It provides a simple API compared to Selenium, so the learning curve is not too steep. Developers can quickly get familiar with it and start writing the end-to-end UI tests. You can also take snapshots and compare them using Protractor. It also allows you to run parallel test cases on different machines.

22. What is a test automation platform?

A test automation platform is a tool or a framework that makes it easy to automate software testing. It uses programs and scripts that are written by developers or testers to automate the entire process.

A test automation platform typically provides all the functionality that you would need to start with automated testing. It saves you from using a plethora of tools and makes them work with each other.

Test automation platforms primarily find their use in complex or large software projects where it's difficult or cumbersome to perform manual testing on all the functionality provided by the software.

23. What are some of the alternatives to Selenium?

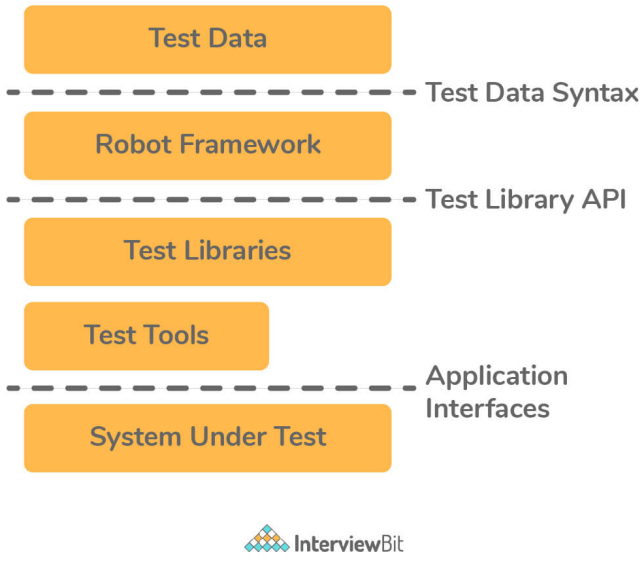
For a long time, Selenium has been one of the most popular test automation tools preferred by many teams. However, it is a very sophisticated tool with a steep learning curve, and it might not be suitable for all test projects. In recent years some popular alternatives have emerged, listed below.

- **Cucumber**
  - It's an open-source testing tool that allows writing tests in a simple, plain language readable by anyone on the team.
  - It focuses on behavior-driven development, where the human-readable description of the functionality is used as the basis for testing.
- **Cypress**
  - Cypress is a free and open-source testing tool. It's written in JavaScript and has become very popular recently because of its simplicity and advanced features that allow you to perform browser testing.
  - Cypress makes it easy to write and debug unit tests, end-to-end tests, integration tests. It also supports taking snapshots and recordings to help to reproduce the bugs.
- **Robot Framework**
  - Robot Framework is a generic open-source automation framework. It can be used for test automation and robotic process automation (RPA).

24. What is the Robot framework? Provide a brief overview of its architecture.

Robot Framework is an increasingly popular, open-source automation testing framework primarily used for robotic process automation (RPA). Robotic process automation tries to emulate human actions and interactions with software. Similar to real humans, robotic automation can understand (to an extent) what is on screen, press keys and buttons, navigate to links and extract data.

The Robot Framework is written in Python and operating system agnostic. Most of the libraries in the ecosystem are also open-source. It has a modular architecture open for extension with other libraries. It defines the test data in files using a special syntax that's specific to the framework. A test suite contains multiple such tests.



When you start the execution of the tests, the Robot Framework parses the test data and uses the keywords provided by the libraries to interact with the software. These libraries communicate with the software directly or indirectly using driver tools.

The Robot Framework runs the test from the command line. However, you can get detailed reports and logs in both XML and HTML formats. The framework has good support for standard libraries out-of-box, e.g. ArchiveLibrary, Browser Library, DataDriver Library, HttpRequestLibrary (for Java), etc.

25. What are the test library APIs provided by the Robot Framework?

The Robot Framework has three test library APIs.

- **Static API**: A module or a class containing methods map directly to the keyword names that take the same arguments as the implementing methods.
- **Dynamic API**: The names of the keywords to implement and how to implement is determined at runtime.
- **Hybrid API**: Combination of both the static and dynamic API. Libraries are classes containing the methods that tell them which keywords to implement, but those keywords must be available directly.

26. How will you automate the basic login in a web application?

Assuming a tester has configured the test environment and a test tool like Selenium, here are the steps I would take to automate the login functionality.

- Test the login manually to understand all the input fields, checkboxes, and buttons on the login screen. Keep a note of which pages the user is redirected to in both successful and failed logins.
- Prepare a test dataset that contains the username and password combinations. The inputs consist of varying lengths and have alphanumeric character sets.
- Develop test cases to test various paths the user might take in a real-world scenario. Note down the expected outputs for each test case.
- In the test tool, configure each test case to be manually invoked, and use the test data prepared in step 2. Record the instances where the actual output doesn't match the expected result.
- Verify and validate the success/error messages and the redirects after each login attempt.

27. What are some risks associated with automated testing?

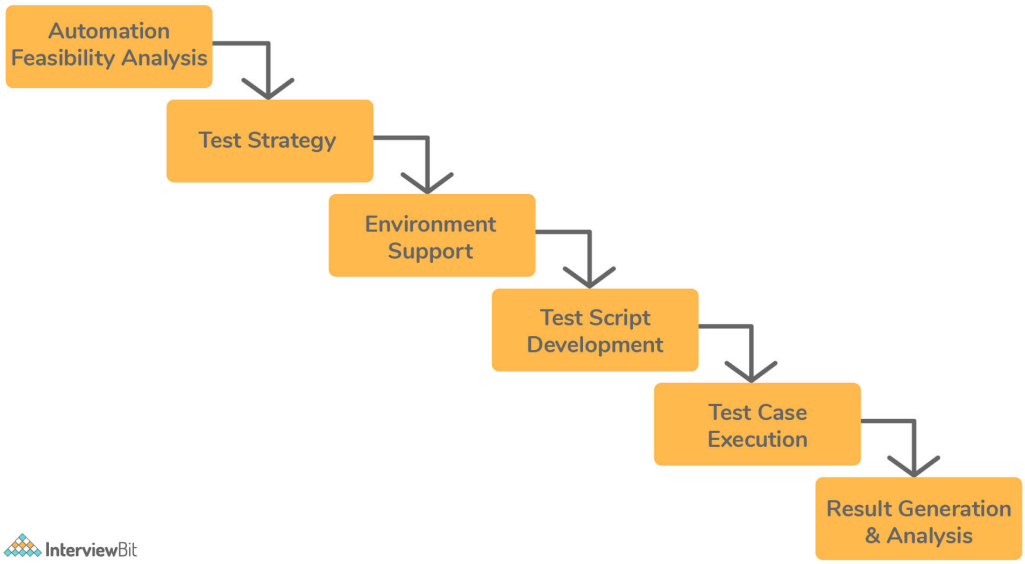
Although test automation comes with benefits such as efficient and fast, repeatable tests, there are a few risks a team should be aware of.

- **Negative ROI**
  - A team can make a considerable investment to get automation testing going. Automated tests require lots of code and expensive tools. Developers and testers spend significant time in learning and implementing automated tests.
  - However, as it's put into practice, the team might realize that the testing strategy is not providing any real value, as the software is complex with constantly changing configuration and features. They have to keep the tests constantly up-to-date.
- **Playing catch-up with the technology.**
  - As with any software, automation testing tools and frameworks go through constant evolution. There is a steep learning curve to the automation tools and need prior programming experience.
  - Instead of picking a tool and using it well, the development/testing team spends their time learning and playing with different tools and technologies. In that case, the automation testing might not realize its original promised value.
- **Maintenance Risk**
  - All code has to be maintained and kept up-to-date with the changing requirements and fixing bugs. Test automation code is no exception to that.
  - Instead of building new features and fixing bugs in the software, if developers and testers find themselves spending most of their time working on the automation framework, automated testing has failed.

28. What are the different phases in an automation testing life cycle?

Similar to a software development and software testing life cycle, automation testing has its life cycle. Here are the major phases that an automation testing project goes through.

- Figure out the scope of automation testing
- Choose the correct automation frameworks and tools
- Design a test plan + test execution strategy
- Set up the test environment
- Development and execution of the test cases
- Analysis and generation of test reports



29. What is CAPTCHA?

CAPTCHA stands for Completely Automated Public Turing Test to tell Computers and Humans Apart. It is a type of security measure and is also known as challenge-response authentication.

The primary goal of the CAPTCHA is to protect you from spam or denial-of-service attacks by bots/scripts by asking you to complete a simple test that is difficult for computers to follow. It proves you are human and not a computer.

A CAPTCHA challenge consists of two parts:

- A randomly generated sequence of letters/numbers. These alphanumeric characters appear as distorted images and might appear behind other non-significant characters to make it difficult for a computer to parse them.
- A text box where the user is supposed to copy the characters. To pass the test and prove that they are human, the user types the characters in the text box.

30. How do you automate the testing of CAPTCHA?

It's not possible to automate the testing of CAPTCHA. That is the goal behind any good CAPTCHA strategy. By definition, a computer can't automate it. If it could, then it's not a good challenge that you can use in your application.

However, if you need to test an application that uses CAPTCHA, you have to work with the development team to build a workaround or a back door that allows the automated test to bypass the CAPTCHA challenge. It's important to restrict this workaround only in the test environment and not release it to production.

31. What are some development practices to follow when writing automated tests?

All the software development rules apply when writing automated tests. Here are some of the best practices that one can apply for tests.

- **Validating the tests will fail**
  - Similar to testing that software will work, it's essential to ensure the test will fail if the feature its testing doesn't meet the criteria.
  - A test that never fails is worse than having no tests, as it gives a false assurance that the feature is working.
- **Don't Repeat Yourself (DRY)**
  - Avoiding duplication in code is crucial.
  - The benefit of this strategy is that the change is isolated to a single location, preventing bugs and errors.
- **Keep Functions Small**
  - Since the testers who are creating automated tests are not familiar with good coding standards, it's easy to fall into the trap of creating giant functions that try to do everything.
  - This quickly results in unmaintainable code that the team is afraid to touch when the requirements change, resulting in out-of-date tests that test the legacy behavior of the system.
- **Write Good Documentation**
  - Having well-written documentation explains not only the what, but then why is important for new team members when they try to understand the tests.
  - It can also help the person who wrote the tests when they try to modify/understand the tests in the future.

32. When selecting an automation tool, what features will you look for?

Here are some of the features to look for when selecting an automation tool:

- Test Environment support,
- Ease of use,
- Debugging features,
- Testing capabilities for various elements,
- UI element identification features,
- Should allow database testing.,
- Should support multiple frameworks.

Conclusion

33. Conclusion

Software testing is an important activity that ensures quality, giving the confidence to release the software to customers. Automation testing is a type of software testing where the tests are automated using tools, scripts, and frameworks, which improves the efficiency and speed of testing. This article explained automation testing and its importance in software development. It also explained different types of tools used in automation testing such as Selenium.

However, automation testing is only a single component of a good software testing strategy. Good old exploratory and manual testing conducted by a human tester are still crucial to shipping a quality software product. As a long-term strategy, the best way to improve the testing process is to test frequently, measure the results, gather feedback and use it to get better.

References:

- Test Pyramid: Fowler, Martin.
  https://martinfowler.com/bliki/TestPyramid.html
- Selenium, an automated testing framework
  https://www.selenium.dev/documentation/
- Protractor, a browser automation testing tool
  https://www.protractortest.org/
- AngularJS, a JavaScript framework
  https://angularjs.org/
- Cucumber, a BDD testing, and collaboration tool
  https://cucumber.io/
- Cypress, a browser automation tool
  https://www.cypress.io/
- Robot Framework, an open-source automation framework
  https://robotframework.org/

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions