Selenium Interview Questions and Answer(Infosys 3+ SDET)

1 . Tell me about your roles and responsibilities in Project. *(Disclaimer: You must tailor this to your actual experience. The following is a standard template for a 3-year SDET/Automation engineer).*

"In my current project, I work as an Automation Test Engineer working within an Agile Scrum team. My primary responsibilities include:

1. **Test Planning & Strategy:** Participating in sprint planning and grooming sessions to understand user stories and identify candidates for automation versus manual testing.
2. **Framework Maintenance:** I am actively involved in maintaining and enhancing our hybrid automation framework (Selenium + Java + TestNG + Maven). This involves updating page objects when the UI changes and creating common utility functions.
3. **Script Development:** I develop robust, data-driven test scripts for regression suites based on accepted criteria.
4. **Execution & Reporting:** I manage the execution of nightly regression suites via Jenkins CI/CD pipelines. Every morning, I analyze the TestNG/Extent reports, triage failures as application bugs or script issues, and report them in Jira.
5. **API Testing:** I also perform API validation using RestAssured for backend services before the UI is integrated.
6. **Code Review:** Reviewing the automation code merged by junior team members to ensure coding standards are followed."

2 . Have you created framework from scratch ? Explain architecture. *(At 3+ years, you are expected to understand the architecture deeply, even if you didn't build the very first line).*

"Yes, I have been heavily involved in designing and setting up framework components from scratch. We use a **Hybrid Framework** based on the **Page Object Model (POM)** design pattern.

The architecture consists of:

1. **Base Class:** Before/After methods to initialize WebDriver, manage browser capabilities, and handle tear-down.

2. **Pages Layer (POM):** Separate Java classes for each web page, containing WebElements (locators defined using @FindBy) and methods to interact with them. This ensures ease of maintenance.
3. **Test Layer:** TestNG classes containing actual test scenarios (@Test methods) containing assertions. These scripts rely on Page objects.
4. **Utilities Layer:** Common helper classes for reading from Excel (Apache POI) or properties files, generic explicit wait methods, and screenshot capture utilities.
5. **Configuration:** pom.xml for managing Maven dependencies and testng.xml for managing test suites and parallel execution controls.
6. **Reporting:** We integrate Extent Reports via TestNG listeners for detailed graphical reports.
7. **CI/CD:** The entire suite is triggered via a Jenkins job using the Maven Surefire plugin."

3 . Which design pattern you have used in framework ? "The primary design pattern is the **Page Object Model (POM)**. It helps separate page-specific locators and actions from the test logic, making the code readable and maintainable. Besides POM, we also use:

- **Singleton Pattern:** To ensure only one instance of the WebDriver runs per thread during parallel execution.
- **Factory Pattern:** Sometimes used if we need to spin up drivers for different browsers dynamically."

4 . Difference between Comparable and Comparator ?

- **Comparable:** It is an interface from java.lang package used to define the *natural ordering* of objects. The class itself implements Comparable and overrides the compareTo() method. You can only have one comparison logic. (e.g., Sorting Students by ID by default).
- **Comparator:** It is an interface from java.util package used to define *custom ordering*. You create separate classes that implement Comparator and override the compare() method. You can have multiple comparators for different sorting logics without touching the original class code. (e.g., Sorting Students by Name, then another Comparator for sorting by Grade).

5 . What are cases where you will use finally block ? "The finally block is used for **resource cleanup**. It executes irrespective of whether an exception occurs in the try block or is caught in the catch block. Common use cases:

1. Closing database connections (JDBC).
2. Closing file input/output streams.
3. Quitting the Selenium WebDriver instance (driver.quit()) at the very end of a test setup to ensure no orphan browsers are left open."

6 .What is 3 Amigos ? "It's an Agile practice where the **Business Analyst (Product Owner)**, the **Developer**, and the **Tester** meet before development begins on a user story. The goal is to ensure a shared understanding of the requirements (the 'What', the 'How', and the 'What if'). This is where we define clear Acceptance Criteria and discuss edge cases early to prevent defects later."

7 . Difference between Scrum and Kanban

- **Scrum:** Structured agile approach. Works in fixed-length iterations called **Sprints** (e.g., 2 weeks). It has defined roles (Scrum Master, PO) and specific ceremonies (Planning, Daily Standup, Review, Retro). Teams commit to a specific amount of work for the sprint.
- **Kanban:** Continuous flow approach. There are no fixed sprints. It focuses on visualizing work on a board and limiting **Work In Progress (WIP)** to improve flow and reduce cycle time. You pull new work only when current work is finished.

8 . Smoke VS Sanity Testing , Regression vs Retesting .

- **Smoke Testing:** Broad and shallow testing done on a new build to ensure critical functionalities work before deep testing begins. It verifies build stability. (Did the build deploy successfully? can I log in?).
- **Sanity Testing:** Narrow and deep testing focused on a specific new feature or bug fix to verify that particular area works correctly.
- **Regression Testing:** Testing existing, unchanged functionalities to ensure that recent code changes (new features or bug fixes) have not broken anything that was previously working.
- **Retesting:** Testing a *specific failed test case* again after the developer has fixed the defect, to verify the fix.

9 . In TestNg how will you ensure that screenshots will be taken only for fail test cases.

"We achieve this using **TestNG Listeners**.

1. Create a class that implements the ITestListener interface.
2. Override the onTestFailure(ITestResult result) method.
3. Inside this method, write the code to capture the Selenium screenshot and save it to a specific folder, usually naming the file after the failed test method name found in the result parameter.
4. Add this listener to the testng.xml file or add the @Listeners annotation above your test class."

10. How to perform data driven testing in postman ?

1. Prepare a data file (CSV or JSON) containing the test data rows with headers matching variable names.
2. In the Postman request, replace hardcoded values with variables using double curly braces, e.g., {{username}} in the request body or URL parameters.
3. Use the **Collection Runner**.
4. Select the collection, upload the data file, and Postman will automatically detect the number of iterations based on the rows in the file.
5. Run the collection; it will execute the requests iteratively using data from each row.

1 1 .What are the important validation you will put on a API. "When validating an API, I check:

1. **Status Code:** The most basic check (e.g., 200 OK for sucess, 201 for created, 400 for bad request).
2. **Response Body Schema:** Does the JSON/XML structure match the contract defined in Swagger/OpenAPI? Are required fields present?
3. **Data Correctness:** Does the data returned in the body match what was expected based on the input parameters or database state?
4. **Response Headers:** Checking Content-Type (e.g., application/json) or security headers.
5. **Response Time:** Ensuring the API responds within acceptable performance limits."

1 2 . Suppose you have 200 test case and 49 test failed , how would you collect the data of only failed test cases in testNg. Best way ? "The best, built-in way is to use the **testng-failed.xml** file. When TestNG executes, it automatically generates a test-output folder. Inside that folder, it creates a file named testng-failed.xml. This XML only contains the test methods that failed during the last execution. To rerun only the failures, we simply right-click and run this specific XML file."

1 3 .What is maven surefire plugin ? "Maven is primarily a build tool, it doesn't know how to run tests by itself. The **Maven Surefire Plugin** is the plugin used during the test phase of the Maven build lifecycle to execute unit and functional tests (like JUnit or TestNG). We configure this plugin in the pom.xml to specify the location of our testng.xml suite file so that mvn test triggers our Selenium tests."

📌 **Round 2 (Technical - Selenium & Coding)**

1 4 . Explain how do you pick which test cases to automate ? "We don't automate everything. I prioritize based on these criteria:

1. **High Risk/Business Criticality:** Core features that must work for the business to function.
2. **Repetitive Tasks:** Test cases executed frequently, like Smoke tests or every Regression cycle.
3. **Data-Driven scenarios:** Tests that need to be run with multiple sets of data inputs.
4. **Stability:** The feature should be relatively stable and not changing every sprint. *We avoid* automating ad-hoc tests, usability testing, or features that are highly volatile."

1 5 . Write code to find Broken Links in Selenium "The concept is to collect all anchor tags (<a>), extract their href attribute, establish an HTTP connection, and check the response code. Anything >= 400 is considered broken."

Java

```
// Assumes driver is initialized

List<WebElement> links = driver.findElements(By.tagName("a"));


for(WebElement link : links){
```

```
    String url = link.getAttribute("href");

    if(url == null || url.isEmpty()){
        System.out.println("URL is empty or null, skipping");
        continue;
    }

    try {
        HttpURLConnection huc = (HttpURLConnection)(new URL(url).openConnection());
        huc.setRequestMethod("HEAD"); // HEAD is faster as it doesn't download body
        huc.connect();
        int respCode = huc.getResponseCode();

        if(respCode >= 400){
            System.out.println(url + " is a BROKEN link. Code: " + respCode);
        } else {
            System.out.println(url + " is a valid link. Code: " + respCode);
        }
    } catch (IOException e) {
        System.out.println("Error connecting to URL: " + url);
    }
}
```

16. How to you handle multiple frames ? "Selenium can only process elements within the current browsing context. If elements are in an <iframe>, we must switch to it first. We use the driver.switchTo().frame() method. We can switch by:

1. **Index:** driver.switchTo().frame(0); (First frame)

2. **Name or ID:** driver.switchTo().frame("frameName");
3. **WebElement:** Find the frame element first, then switch: driver.switchTo().frame(driver.findElement(By.id("frameId")));"

1 7 .You have 10 links how will you print title of each link using selenium. Give Optimal approach "The naive approach of clicking, getting title, and navigating back is inefficient and often leads to StaleElementReferenceException because the DOM reloads.

The optimal approach is to first grab all the URLs, then iterate through the URLs.

Java

```
// 1. Get all link elements

List<WebElement> linksElements = driver.findElements(By.xpath("//div[@class='link-container']//a"));


// 2. Extract HREFs into a separate list to avoid Stale reference issues later

List<String> urls = new ArrayList<>();

for(WebElement element : linksElements){

   urls.add(element.getAttribute("href"));

}


// 3. Iterate through the URLs, navigate and get title

for(String url : urls){

   driver.navigate().to(url);

   // Optional: Add a wait here if page load is slow

   System.out.println("Page Title: " + driver.getTitle());

}

// Note: If the requirement implies the link text *is* the title,

// we just iterate the WebElements and print .getText(), no navigation needed.

// But usually, this question means the title of the destination page.
```

18. In what cases we need to use implicit / explicit wait

- **implicitWait:** This is a global setting applied to the driver instance. It tells WebDriver to poll the DOM for a certain amount of time when trying to find *any* element if it's not immediately available. It's generally used as a baseline safety net but is not recommended as the primary synchronization strategy because it slows down tests when elements aren't found.
- **explicitWait (WebDriverWait):** This is applied to a specific element for a specific condition. It is the preferred method. We use it when an element takes time to load dynamically, become clickable, or become visible due to AJAX calls.
    - *Example:*
      wait.until(ExpectedConditions.elementToBeClickable(By.id("submitBtn")));

19.What is null pointer exception. "A NullPointerException is a RuntimeException in Java that occurs when the program attempts to use an object reference that has a value of null. This means the reference isn't actually pointing to any object instance in the heap memory, but the code is trying to call a method on it or access one of its fields."

20.Follow up question (check question 3) , how would you switch to parent frame. "To move back from a child frame to its immediate parent frame, we use:
driver.switchTo().parentFrame();

If we want to exit all frames entirely and return to the main page HTML, we use:
driver.switchTo().defaultContent();"

21.Difference between driver.get() vs driver.navigate().to()

- driver.get("url"): This is the standard way to load a new webpage. It waits until the page is fully loaded (specifically, until the document.readyState is "complete") before returning control to the test script.
- driver.navigate().to("url"): This does essentially the same thing as .get() (it also waits for page load). However, it is part of the Navigation interface, which allows chaining other browser history commands like .back(), .forward(), and .refresh().

22. How to handle Location popup in selenium ? "Browser Location popups (like 'wants to know your location - Allow/Block') cannot be handled by standard Selenium findElement commands because they are part of the browser UI, not the web page HTML. We must handle them by configuring browser capabilities *before* launching the browser. For Chrome, we use ChromeOptions to disable geo-location info bars or automatically permit location access for specific sites."

2 3 .How to handle dynamic dropdowns in Selenium ? "Dynamic dropdowns usually don't use the standard <select> tag. They often load content via AJAX when clicked. My approach is:

1. Click the dropdown trigger element.
2. Use an **Explicit Wait** (WebDriverWait with visibilityOfAllElements) to ensure the list of options is fully loaded and visible in the DOM.
3. Find the list of all option elements using findElements.
4. Iterate through the list using a for-each loop, checking the text of each option.
5. When the desired text matches, click that element and break the loop."

2 4 . default vs public access modifier.

- **Public:** A class, method, or variable declared public can be accessed from *any* other class in the Java application, regardless of the package it is in.
- **Default (Package-Private):** If no modifier is specified, it gets the default access. Members are only accessible by classes within the **same package**. They are not visible in different packages, even to subclasses.

2 5 .What is method hiding in Java ? "Method hiding occurs when a child class defines a **static** method with the same signature (name and parameters) as a **static** method in its parent class. Unlike method overriding (which applies to instance methods and supports polymorphism), static methods are bound at compile time. The child's static method 'hides' the parent's static method. Which method gets called depends on the reference type holding the object, not the runtime object type."

2 6 .Write Java code to print all the array elements that appears atleast 2 times. (means 2 or greater than two) "The most efficient approach is to use a HashMap to count frequencies, which gives a time complexity of O(n)."

Java

```
import java.util.HashMap;

import java.util.Map;


public class Duplicates {

   public static void main(String[] args) {
```

```
        int[] arr = {1, 5, 2, 1, 6, 5, 5, 8, 2};

        // 1. Create a map to store number -> count
        Map<Integer, Integer> countMap = new HashMap<>();

        // 2. Iterate array and fill map
        for (int num : arr) {
            if (countMap.containsKey(num)) {
                // If it exists, increment current count by 1
                countMap.put(num, countMap.get(num) + 1);
            } else {
                // If first time seeing it, initialize count to 1
                countMap.put(num, 1);
            }
        }

        System.out.println("Elements appearing at least twice:");
        // 3. Iterate map to check conditions
        for (Map.Entry<Integer, Integer> entry : countMap.entrySet()) {
            if (entry.getValue() >= 2) {
                System.out.print(entry.getKey() + " ");
            }
        }
        // Output should be: 1 2 5
    }
}
```