**Technical Round Questions – Round 1st L1-Technical Round:**

1. Introduce Yourself with your roles and responsibilities as QA.
2. Locating web elements using xpath and CSS selectors.
3. Syntax to write implicit wait and explicit wait at what time you will use?
4. How do you automate dropdown?
5. How do you automate alerts?
6. What is JS executor why it is used? - pending
7. What you’re working on current project as a QA. -
8. Write a programme to find equilibrium element in the mentioned array.

Example: Equilibrium element is the element, sum of its left side elements and sum of its right-side elements are equal.

Int[] arr={1,2,3,1,1,1}; - pending

Output: 3, In this array equilibrium element is 3.

1. How do you Automate Multiple windows explain. - pending
2. Write the test cases to test a Post API with all functional aspects.
3. How do you Automate Post API with Rest Assured please explain
4. Write BDD cucumber feature steps to Automate Post API
5. I want to run Particular test scenarios how do you achieve this in cucumber framework
6. What is difference between static and non-static keywords in Java. – pending
7. Explain how you used inheritance in your current framework. - Pending
8. Have you integrated your framework to Jenkins? If yes How, If not why please explain
9. Write set of git commands to commit your latest changes into your branch
10. Assume you and your friend worked on the same Java class and on same Scripting your friend is committed his code to master, and it is available in master, what will you expect while raising PR how you resolve it.
11. Create a Basic Maven Framework to launch browser and navigate to <https://google.com> and run that framework with maven command.
12. If you raise JIRA developer blamed, you saying it is not defect what will you do?
13. Write a Programme to separate letters and symbols in a string as show below.

Example:

Input – String s= “M@y n#a$m%e i^s K&i^r!a0n”;

Output- String s= “My name is Kiran @#$%^&^!0”;

1. Explain Your framework structure in Note pad file.
2. What is Boundary Value Analysis? Write test cases for a file which accept 10-digit mobile number.
3. How you test a Post request with multiple request input parameters kindly explain.
4. What you mean by status code of series 301,303,……3XX etc
5. You have 5L and 3L cans by using both I want to take only 4L of water. How do you achieve this?
6. How do you drive the data in your framework?
7. Write a Programme to find the occurrence of each character in the Stirng.

Example:

Input - String s = “KiranKiran”;

OutOut ­– K-2

i-2

r-2

a-2

n-2

1. Write any basic program in such a way it should execute even though exception occurs.
2. What you mean by thread in Java
3. What happens if static variable is used in multithreaded environment.
4. How do you handle Frames in Selenium explain
5. Open <https://www.programiz.com/java-programming/online-compiler/> and locate Run button use any locator technique you want.
6. How do you handle Particular element in webpage which takes usually more time than all elements.
7. How do you launch browser in latest selenium version
8. Once you install Java, why you need to mentioned Java path in your computer system variables? explain
9. Write a Programme to rotate a Array Element in right direction by user input.

Example:

Input – int[] a={1,2,3,4}; n=2 (How many time we want to rotate in right direction)

Output- int[] a={3,4,1,2};

## **Example: High Severity, Low Priority Defect**

### 🔧 ****Scenario****

You're testing an **e-commerce website**. During testing, you discover that:

**The "Forgot Password" feature crashes the application completely when used — but only in Internet Explorer 11**, which the company no longer officially supports.

### 🚨 ****Why High Severity?****

* The application **crashes** — that's a **major functional failure**.
* It affects **user experience** and potentially causes **data inconsistency** or system downtime.
* So, **technically**, it’s a **high severity bug**.

### 🕗 ****Why Low Priority?****

* It occurs **only in Internet Explorer 11**, which is a **deprecated and unsupported browser**.
* The business has already moved to support only modern browsers (Chrome, Edge, Firefox).
* There's **no real user impact** because no one is expected to use IE11 anymore.

### ✅ ****Summary****

| **Attribute** | **Value** |
| --- | --- |
| **Severity** | High – app crash |
| **Priority** | Low – outdated browser |
|  |  |

**CHALENGES FACED DURING FRAMEWORK DEVELOPMENT**

## **UI Automation Challenges (Selenium)**

### 1. ****Dynamic Elements / Locators****

* 🔁 IDs or classes change every time the page reloads.
* **Solution**: Use dynamic XPath or CSS selectors, parent-child hierarchy, or contains() functions.

### 2. ****Synchronization Issues****

* Elements not ready when Selenium tries to interact.
* **Solution**: Implement **explicit waits** using WebDriverWait and ExpectedConditions.

### 3. ****Handling Popups / Modals / Alerts****

* Pop-ups can block the next steps.
* **Solution**: Use driver.switchTo().alert() or switch to frames if needed.

### 4. ****Cross-browser Compatibility****

* Script works in Chrome but breaks in Firefox/Edge.
* **Solution**: Use cross-browser testing tools like **Selenium Grid**, **BrowserStack**, or **Playwright**.

### 5. ****Test Data Management****

* Relying on hardcoded or outdated test data.
* **Solution**: Use external data files (Excel, JSON, DB, Faker libs) or mock environments.

### 6. ****Framework Scalability****

* As test cases grow, maintaining them gets tough.
* **Solution**: Use **Page Object Model (POM)**, proper abstraction, and reusable components.

### 7. ****Flaky Tests****

* Tests pass sometimes and fail sometimes.
* **Solution**: Improve wait strategies, retry logic, better environment control.

## 🔗 **API Automation Challenges (RestAssured / Postman / HttpClient)**

### 1. ****Dynamic Data in Requests****

* Auth tokens, timestamps, or IDs change frequently.
* **Solution**: Capture tokens dynamically, use variables or custom scripts to generate data.

### 2. ****Handling Authentication****

* APIs might use OAuth2, JWT, or other methods.
* **Solution**: Create reusable auth utility methods to fetch and store tokens.

### 3. ****Chaining Requests****

* When one request's output is needed in another.
* **Solution**: Use global variables or pass values between methods.

### 4. ****Data Validation****

* Complex nested JSON makes assertion difficult.
* **Solution**: Use JSONPath or deserialization with POJOs (in Java).

### 5. ****Versioning & Backward Compatibility****

* API changes break older tests.
* **Solution**: Maintain versioned tests and include version headers or URL paths.

### 6. ****Mocking/Service Unavailability****

* Backend might not be available during test cycles.
* **Solution**: Use mocking tools like **WireMock**, **MockServer**, or **Postman Mock Servers**.

### 7. ****Rate Limiting / Throttling****

* Tests fail when rate limits are hit.
* **Solution**: Add delay or batch processing in test runs.

## 💡 General Best Practices

* ✅ Integrate UI and API in your framework for E2E testing.
* ✅ Use logging and reporting (like **ExtentReports**, **Allure**, etc.).
* ✅ Use CI/CD tools (like **Jenkins**, **GitHub Actions**) for execution.
* ✅ Modularize utilities (auth, config, logger, data provider).

Framework Interview Questions Scenario Based – Selenium with Java Maven

**1. Framework Design and Structure:**

**Q1:** *You are tasked with designing an automation framework using Selenium, Maven, and Java. Can you explain the structure of your framework? What kind of design patterns would you use and why?*

**Answer:**  
The framework can be structured using a modular approach, where each component has a clear responsibility. A typical structure might look like this:

* **src/main/java**
  + com.project.pages: Contains Page Object Model (POM) classes. Each page of the application will have a corresponding class.
  + com.project.utils: Contains utility classes like WebDriver setup, waits, and configuration management.
  + com.project.testcases: Contains test classes where the actual test cases are written using TestNG.
* **src/test/resources**
  + config.properties: Stores configurations like browser types, URLs, etc.
  + testdata.xlsx: Stores test data in Excel for parameterized tests.

The **Page Object Model** (POM) design pattern is ideal for test automation as it encourages the reuse of code and provides a separation of concerns. This improves maintainability and scalability of the test suite.

**Q2:** *In your Selenium Maven Java framework, how would you handle different test environments (e.g., dev, staging, prod)? How can Maven help in managing different environments?*

**Answer:**  
We can manage different environments using the config.properties file. We can create different config-dev.properties, config-staging.properties, and config-prod.properties files for each environment. Based on the environment passed during the test run, we can load the corresponding properties file using Java’s Properties class.

Maven helps by passing environment-specific profiles through the pom.xml. We can specify properties in different profiles and pass the profile name when running the tests with mvn test -P <profile\_name>.

**Q3:** *You are working with a team where multiple automation scripts are running in parallel using Maven. How would you ensure that the tests run in isolation and avoid test data conflicts?*

**Answer:**  
We can ensure test isolation by:

1. **Using unique test data for each test** (e.g., creating random or unique test data for each execution).
2. **Using TestNG’s @BeforeMethod and @AfterMethod** annotations to reset the test environment before and after each test method.
3. **Configuring parallel test execution in TestNG** by setting <suite parallel="methods" thread-count="5"> in the testng.xml to ensure methods are executed in isolation.
4. **Using @DataProvider** in TestNG for parameterized tests, which provides different sets of data for each test method.

**2. Handling Test Data:**

**Q4:** *You have a requirement where the test data varies based on different environments (e.g., dev, qa, staging). How would you manage this in your Selenium Maven Java automation framework?*

**Answer:**  
We can manage environment-specific test data by using **properties files** for each environment (e.g., data-dev.properties, data-staging.properties). The properties files will contain test data like usernames, passwords, and URLs.

At runtime, the corresponding properties file can be loaded based on the environment using the System.getProperty("env") method and Java's Properties class to read the values dynamically.

**Q5:** *You need to automate a scenario where you are testing multiple sets of data in a loop (e.g., logging in with different users and credentials). How would you handle this in your framework using TestNG and Maven?*

**Answer:**  
We can use **TestNG's @DataProvider** to pass multiple sets of data to a single test method. The test will loop through the data provided by the @DataProvider and execute the test case for each data set.

Example:

java

Copy

@DataProvider(name = "loginData")

public Object[][] getData() {

return new Object[][] {

{"user1", "password1"},

{"user2", "password2"},

{"user3", "password3"}

};

}

@Test(dataProvider = "loginData")

public void testLogin(String username, String password) {

// Automation steps for login using username and password

}

**3. Dependency Management and Build Process:**

**Q6:** *How would you handle dependencies in your Selenium Maven Java project? What is the importance of the pom.xml file, and how does it help in managing dependencies for your automation framework?*

**Answer:**  
The **pom.xml** file is crucial in Maven projects as it handles all dependency management. Dependencies for libraries like Selenium, TestNG, WebDriverManager, etc., are declared in the pom.xml. When Maven runs, it downloads the required libraries and resolves the dependencies automatically.

Example:

xml

Copy

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**Q7:** *Your framework uses Maven for builds. If a new version of Selenium is released and you need to update the version in your framework, how would you update the dependencies in your Maven project? How would you ensure that the upgrade doesn't break existing tests?*

**Answer:**  
To update the Selenium version:

1. Modify the version number in the pom.xml file.
2. Run mvn clean install to download the updated version.
3. After upgrading, I would **run all existing tests** to check for any breaking changes due to the new version.
4. **Check the Selenium release notes** to ensure there are no major changes that would affect existing functionality.

To prevent breaking changes, I would maintain a **version control system** (e.g., Git) to track and roll back changes if needed.

**Q8:** *Explain the process of integrating Jenkins with your Selenium Maven Java framework. How would you configure it to trigger tests on code commits or on a schedule?*

**Answer:**  
To integrate Jenkins:

1. **Install necessary plugins** (e.g., Maven, Git) on Jenkins.
2. **Create a Jenkins job** and configure the source code repository (GitHub, GitLab, etc.).
3. **Configure the build step** in Jenkins to run mvn clean test for Maven-based projects.
4. Set up **Post-build actions** to **publish test results** (e.g., TestNG reports).
5. To trigger tests on code commits, set up a **webhook** from the repository to Jenkins.
6. For scheduled execution, configure a **Cron job** in Jenkins to run tests at specific intervals.

**4. Parallel Test Execution:**

**Q9:** *You need to run your Selenium tests in parallel across multiple browsers. How would you configure your Selenium Maven Java framework to execute tests in parallel on multiple browsers (e.g., Chrome, Firefox, IE)?*

**Answer:**  
We can run tests in parallel by configuring TestNG and Selenium:

1. **In testng.xml**, specify the parallel execution attribute:

xml

Copy

<suite name="Parallel Suite" parallel="tests" thread-count="2">

<test name="Chrome Test">

<parameter name="browser" value="chrome" />

<classes>

<class name="TestClass" />

</classes>

</test>

<test name="Firefox Test">

<parameter name="browser" value="firefox" />

<classes>

<class name="TestClass" />

</classes>

</test>

</suite>

1. In your test class, use @BeforeMethod to instantiate the correct WebDriver based on the browser parameter.

**Q10:** *Can you explain how you would configure your Maven pom.xml file to handle parallel test execution and integrate it with a CI tool like Jenkins for continuous testing?*

**Answer:**  
In **Maven**, we would configure **Surefire plugin** in pom.xml to run tests in parallel:

xml

Copy

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

<configuration>

<parallel>methods</parallel>

<threadCount>4</threadCount>

</configuration>

</plugin>

</plugins>

</build>

In Jenkins, create a job and configure it to run mvn clean test to trigger parallel test execution during the build.

**5. Reporting and Logs:**

**Q11:** *After executing your automation tests, you need to generate detailed reports for the team. How would you generate and customize HTML or XML reports using TestNG in your Selenium Maven framework?*

**Answer:**  
TestNG generates an **HTML report** by default after the tests are executed. To customize the report:

1. Use testng.xml to set the report format:

xml

Copy

<suite name="Suite" verbose="1" parallel="tests">

<listeners>

<listener class-name="org.uncommons.reportng.HTMLReporter"/>

</listeners>

</suite>

1. For detailed customization, implement a **custom TestNG listener** that will generate reports in your preferred format.

**Q12:** *You noticed that your Selenium tests occasionally fail due to timing issues, such as waiting for an element to appear on the page. How would you handle synchronization issues in your framework? What strategies would you use for implicit and explicit waits in Selenium?*

**Answer:**  
I would use **Explicit Waits** with WebDriverWait to wait for specific conditions like element visibility, clickability, or presence. For example:

java

Copy

WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementId")));

**Implicit Waits** can be set globally for a timeout period when an element is not immediately found.

java

Copy

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Explicit waits are preferred over implicit waits for handling specific timing issues.

**6. Exception Handling and Debugging:**

**Q13:** *During test execution, your Selenium script encounters an exception due to a missing element or incorrect locator. How would you handle this exception in your framework to ensure that the test fails gracefully without affecting the execution of other tests?*

**Answer:**  
I would catch the exception using try-catch blocks. Inside the catch block, log the exception and take a screenshot for later debugging. This way, the script can continue with other tests.

Example:

java

Copy

try {

WebElement element = driver.findElement(By.id("nonExistent"));

} catch (NoSuchElementException e) {

System.out.println("Element not found: " + e.getMessage());

takeScreenshot();

}

**Q14:** *A test fails intermittently in a specific browser but works fine in others. How would you troubleshoot this issue in your Selenium Maven Java framework? What debugging tools or techniques would you use?*

**Answer:**  
I would first ensure that the test is not dependent on a specific browser version or WebDriver implementation. Then, I would:

1. **Review the browser logs** and **Selenium logs** for any error messages.
2. **Check the WebDriver version** to ensure compatibility with the browser.
3. **Run the test in headless mode** to rule out UI-related issues.
4. Use **browser-specific capabilities** to set the desired browser configuration.

Framework Interview Questions Scenario Based – API Rest Assured with Java Maven

**1. Framework Design and Structure:**

**Q1:** *You are tasked with designing a Rest Assured-based API automation framework using Maven and Java. Can you explain the structure of your framework and the design patterns you would use?*

**Answer:**  
A modular approach for organizing the framework would be ideal, with the following structure:

* **src/main/java**
  + com.project.api: Contains API-related classes, such as GET, POST, PUT, DELETE request methods, and common response validation methods.
  + com.project.utils: Contains utility classes like Request/Response builders, API authentication methods, and logging utilities.
  + com.project.testcases: Contains test cases or tests for different API endpoints written in TestNG/JUnit.
* **src/test/resources**
  + config.properties: Stores configurations like API base URL, authorization tokens, etc.
  + data.json: Stores request payloads for POST, PUT, etc.

**Design Pattern:** I would use the **Page Object Model (POM)** design pattern in API testing, which is also referred to as **Service Object Model** in API testing. This pattern promotes code reuse, maintains clean code, and allows easy changes to the request/response structure without affecting the test scripts.

**Q2:** *How would you handle environment-specific configurations (e.g., development, staging, production) in your Rest Assured framework using Maven?*

**Answer:**  
You can manage environment-specific configurations by creating different **properties files** for each environment (e.g., config-dev.properties, config-prod.properties). You can use the Properties class to load the appropriate configuration file based on the environment set at runtime (e.g., -Denv=dev).

Using Maven, you can configure **profiles** in the pom.xml file to pass environment-specific configurations during the build:

xml

Copy

<profiles>

<profile>

<id>dev</id>

<properties>

<baseUrl>https://dev-api.example.com</baseUrl>

</properties>

</profile>

<profile>

<id>prod</id>

<properties>

<baseUrl>https://api.example.com</baseUrl>

</properties>

</profile>

</profiles>

**Q3:** *Your API automation tests need to interact with various authentication mechanisms (e.g., Basic Authentication, OAuth, API Keys). How would you manage different authentication types in your framework?*

**Answer:**  
For managing different authentication mechanisms:

1. **Basic Authentication:**  
   Use Rest Assured's auth() method to send username and password.

java

Copy

given().auth().basic("username", "password")

1. **Bearer Token Authentication (OAuth 2.0):**  
   Include the token in the request header:

java

Copy

given().header("Authorization", "Bearer " + token)

1. **API Key:**  
   Pass the API key in the request header or query parameter.

java

Copy

given().header("x-api-key", apiKey)

I would implement a utility method to handle different types of authentication and encapsulate the logic in a reusable function.

**2. Test Execution and Data Management:**

**Q4:** *You need to test multiple API endpoints (e.g., GET, POST, PUT, DELETE) with different input data sets. How would you handle data-driven API testing in your Rest Assured framework?*

**Answer:**  
To handle data-driven testing, we can use **TestNG's @DataProvider** or **JUnit’s @ParameterizedTest** to pass different sets of data for each test method. Each test case will execute for every data set provided.

Example with TestNG:

java

Copy

@DataProvider(name = "userData")

public Object[][] getUserData() {

return new Object[][] {

{ "user1", "password1" },

{ "user2", "password2" }

};

}

@Test(dataProvider = "userData")

public void testLogin(String username, String password) {

given().param("username", username).param("password", password)

.when().post("/login")

.then().statusCode(200);

}

**Q5:** *You have to send different types of payloads (e.g., JSON, XML) to your API in various tests. How would you handle dynamic payload creation in your Rest Assured framework?*

**Answer:**  
For dynamic payload creation, I would use **Jackson** or **Gson** for JSON processing and **JAXB** for XML processing. This approach allows creating POJOs (Plain Old Java Objects) that can be converted to JSON/XML dynamically based on the test data.

For example, using Jackson:

java

Copy

ObjectMapper mapper = new ObjectMapper();

MyObject object = new MyObject("John", "Doe");

String jsonPayload = mapper.writeValueAsString(object);

given().body(jsonPayload).when().post("/users").then().statusCode(201);

For **parameterized payloads**, you can read the payloads from JSON or XML files and modify them at runtime based on the test case.

**3. Error Handling and Debugging:**

**Q6:** *Your tests occasionally fail due to network timeouts or server unavailability. How would you handle such errors in your framework?*

**Answer:**  
To handle network timeouts or server unavailability, I would use the **try-catch** blocks for exceptions like SocketTimeoutException, ConnectException, etc., and implement retries.

Example of retry mechanism:

java

Copy

int retries = 3;

while (retries > 0) {

try {

given().get("/api/endpoint").then().statusCode(200);

break; // Break if test passes

} catch (Exception e) {

retries--;

if (retries == 0) {

throw e; // Fail the test after max retries

}

}

}

I would also ensure proper logging of the exception messages and time of failure for better debugging.

**Q7:** *How would you troubleshoot an API test that passes intermittently in your framework?*

**Answer:**  
Intermittent test failures could be due to timing issues, server load, or API rate-limiting. I would:

1. **Check response times** to see if the server is under heavy load.
2. Add **explicit waits** in the API tests (e.g., wait for the resource to become available).
3. Implement **retry logic** for flaky tests.
4. **Log the full API response**, including headers and body, to analyze discrepancies.
5. **Check the API rate limits** and ensure the tests aren’t hitting rate limits by spacing out requests.

**4. API Response Validation and Reporting:**

**Q8:** *How would you validate the response body, status code, and headers of an API response in your framework?*

**Answer:**  
You can validate API responses using **Rest Assured's built-in methods**. For example:

1. **Status Code:**

java

Copy

given().get("/endpoint").then().statusCode(200);

1. **Response Body:**

java

Copy

given().get("/endpoint").then().body("key", equalTo("value"));

1. **Headers:**

java

Copy

given().get("/endpoint").then().header("Content-Type", "application/json");

Additionally, for complex response bodies (JSON/XML), we can use **JSONPath** or **XPath** to validate specific elements:

java

Copy

given().get("/endpoint").then().body("user.name", equalTo("John"));

**Q9:** *You need to generate a report after running your API tests in Rest Assured. How would you generate and customize a test report for API automation using TestNG or any other tool?*

**Answer:**  
We can use **TestNG's built-in reporting** for API tests. TestNG generates an HTML report by default after test execution. To customize this:

1. Configure **TestNG listeners** (e.g., HTMLReporter, XMLReporter) for custom reports:

xml

Copy

<listeners>

<listener class-name="org.uncommons.reportng.HTMLReporter"/>

</listeners>

1. For more detailed reports, I would integrate **Allure** or **ExtentReports** for enhanced visual reporting, which provides more insights into test execution, pass/fail details, and screenshots.

For **Allure Reports**:

xml

Copy

<dependency>

<groupId>io.qameta.allure</groupId>

<artifactId>allure-testng</artifactId>

<version>2.14.0</version>

</dependency>

**5. Advanced API Testing Scenarios:**

**Q10:** *How would you handle an API response with pagination, where the response returns data in chunks (e.g., 100 records per page)?*

**Answer:**  
For pagination, we can implement a loop that iterates through all pages by checking the **next page link** or **page number** in the response.

Example:

java

Copy

int page = 1;

boolean hasNextPage = true;

while (hasNextPage) {

Response response = given().param("page", page).get("/api/items");

hasNextPage = response.jsonPath().getBoolean("nextPageAvailable");

page++;

// Validate response data here

}

**6. Integration with Other Tools:**

**Q11:** \*You need to integrate your Rest Assured API tests with a CI/CD pipeline. How would you set up Jenkins to trigger your API automation tests on code commit

Framework Interview Questions Scenario Based – Performance JMeter

**1. Framework Design and Setup**

**Q1:** *You are tasked with creating a performance testing framework using JMX. How would you structure the framework, and what tools would you include for better test execution, reporting, and scalability?*

**Answer:**  
The performance testing framework can be structured as follows:

* **Test Scripts Directory:**
  + Contains JMX files (one for each service or feature being tested).
* **Test Data Directory:**
  + Stores any CSV files or configuration files for test data (e.g., usernames, passwords, test inputs).
* **Log/Reports Directory:**
  + Stores logs and reports generated from JMeter.
* **Utils Directory:**
  + Contains utility classes for pre/post-processing (e.g., processing JMeter logs, generating dynamic variables for tests).

**Tools to include:**

* **JMeter** for load testing.
* **Jenkins** for continuous integration.
* **Apache Maven** for dependency management and build automation.
* **Grafana/InfluxDB** for real-time monitoring and data visualization.
* **JMeter Plugins** for advanced reporting (e.g., JMeter Plugins Manager).

**Design Considerations:**

* Modular JMX files to test different APIs, web applications, or services separately.
* A parameterized approach to test various environments (Dev, Staging, Production) using property files.
* Automation of test execution with a CI tool like Jenkins to trigger tests after each commit or on a scheduled basis.
* Integration with reporting tools like **Allure**, **Extensive JMeter Reports**, or **Grafana** for enhanced reporting and analysis.

**2. Test Execution and Data Management**

**Q2:** *How would you create a performance testing strategy to handle scenarios where you need to simulate a variety of user behaviors, such as login, browsing, and checkout actions on an e-commerce website?*

**Answer:**  
To simulate user behaviors effectively, I would use the **Thread Groups** and **Samplers** in JMeter:

1. **Thread Groups:** Represent virtual users (VUs) simulating different behaviors:
   * **Login Simulation:** Create a thread group to simulate multiple users logging in.
   * **Browsing Simulation:** Create another thread group to simulate users browsing product pages.
   * **Checkout Simulation:** Create a final thread group to simulate users adding items to the cart and completing a purchase.
2. **Timers:** Add **Constant Timer**, **Gaussian Random Timer**, or **Uniform Random Timer** to simulate realistic delays between actions.
3. **CSV Data Set Config:** Use a CSV file to input dynamic data (e.g., user credentials, item IDs) to simulate different login and checkout scenarios. This can be done using **CSV Data Set Config** in JMeter.
4. **Assertions:** Validate server responses using **Response Assertions** (e.g., check that the response contains the expected text or status code).

**Q3:** *How would you implement parameterization in JMeter for a test that requires dynamic input (e.g., different login credentials for each virtual user)?*

**Answer:**  
To implement parameterization in JMeter:

1. **CSV Data Set Config**: This component allows you to feed data into your test scripts from an external CSV file. The CSV file can contain multiple rows of input data (e.g., usernames, passwords).

Example CSV file:

csv

Copy

username,password

user1,pass1

user2,pass2

user3,pass3

In JMeter, add a **CSV Data Set Config** element and point it to the CSV file:

* **Variable Names**: username, password
* **Filename**: path to the CSV file

Now, the **HTTP Request Sampler** for login will use these values dynamically:

plaintext

Copy

Username: ${username}

Password: ${password}

**3. Performance Test Execution**

**Q4:** *You need to run a load test to simulate 1000 virtual users hitting an API simultaneously. How would you set up and execute this test in JMeter?*

**Answer:**

1. **Thread Group Setup**:
   * Add a **Thread Group** to simulate 1000 users.
   * Set **Number of Threads (Users)** to 1000.
   * Set the **Ramp-up period** to define how quickly these users will be created (e.g., 1000 users in 10 minutes = 600 seconds ramp-up).
   * Set **Loop Count** to specify how many times each user should repeat the actions.
2. **Add Samplers**:
   * Add the **HTTP Request Sampler** to simulate the API requests (e.g., GET, POST, PUT requests).
   * Configure the **Server Name or IP**, **Path**, and **Method** for the request.
3. **Add Timers**:
   * Use **Gaussian Random Timer** to simulate realistic user think time.
4. **Listeners**:
   * Add **View Results Tree** (for debugging) and **Summary Report** or **Graph Results** (for final results).
5. **Execution**:
   * Start the test and monitor the results in real-time through JMeter’s GUI or integrate it with **Grafana**/ **InfluxDB** for continuous performance monitoring.
   * Analyze the test results for response times, throughput, and errors.

**Q5:** *How would you analyze the performance test results in JMeter after executing a large-scale load test (e.g., with 1000 users)? What metrics would you focus on?*

**Answer:**  
After executing a large-scale load test, I would analyze several key performance metrics to determine if the application can handle the load:

1. **Response Time**:
   * Check the **average response time**, **95th percentile response time**, and **maximum response time** to ensure the application responds within acceptable limits.
2. **Throughput**:
   * Throughput measures how many requests per second the system can handle. This is important for evaluating system scalability.
3. **Error Rate**:
   * Look for **failed requests** to identify any issues with the application under load. Analyze the **error responses** to determine their cause.
4. **Response Codes**:
   * Examine the HTTP status codes returned (200, 500, 404) to identify server errors or failed transactions.
5. **CPU and Memory Usage**:
   * If integrating with monitoring tools like **Grafana** or **InfluxDB**, check for server-side resource utilization metrics such as CPU, memory, and network usage.
6. **Peak Load Analysis**:
   * Identify the **load at which performance degradation starts** and the **point of failure**.
7. **Visual Graphs**:
   * Use JMeter's **Graph Results** or **Graphana Dashboards** to visualize trends over time for better understanding and quicker decision-making.

**4. Troubleshooting and Optimization**

**Q6:** *During a load test, you notice that the response times for certain API requests are increasing exponentially as the load increases. What steps would you take to troubleshoot and optimize the performance?*

**Answer:**

1. **Check for Bottlenecks**:
   * Look for database queries, slow services, or any components in the stack that might be creating a bottleneck.
2. **Server Resource Usage**:
   * Monitor server-side metrics such as **CPU**, **RAM**, **Disk I/O**, and **Network Usage**. High resource utilization might be causing delays.
3. **Application Logs**:
   * Check server logs for any errors or warnings that might indicate performance issues (e.g., timeouts, server crashes, slow queries).
4. **Test Configuration**:
   * Verify that the load test is configured correctly. Check for issues like **too many virtual users** for the server’s capacity or **incorrect ramp-up time** causing a sudden surge in traffic.
5. **Optimize Load Distribution**:
   * If the application is scaling poorly, try **load balancing** or distributing traffic across multiple servers.
6. **Increase System Resources**:
   * For critical APIs, consider adding **caching** mechanisms (e.g., Redis) to reduce load on the backend systems.
7. **Database Optimization**:
   * Optimize queries or use indexing for slower database queries that might be affecting the response time.

**Q7:** *How would you implement the concept of* ***Distributed Testing*** *in JMeter when your load test needs to simulate millions of users from multiple machines?*

**Answer:**  
**Distributed testing** in JMeter allows you to execute load tests across multiple machines to simulate a larger load:

1. **Set Up JMeter on Multiple Machines**:
   * Install JMeter on several machines (master-slave setup).
2. **Configure the Master (Client) Machine**:
   * Start the JMeter client on the master machine and configure the **Remote Start** feature in the **Test Plan**.
3. **Configure Slave (Server) Machines**:
   * Start JMeter on each slave machine and configure them to communicate with the master machine. You can do this by specifying the IP addresses of the slave machines in the **jmeter.properties** file.
4. **Execute the Test**:
   * Start the test from the master machine. The load will be distributed across all the connected slave machines.
5. **Monitor and Aggregate Results**:
   * Use **JMeter's Distributed Mode** to aggregate results from all slave machines into a single report.

**5. Reporting and Analysis**

**Q8:** *After executing a performance test, how would you generate and customize test reports in JMeter?*

**Answer:**

1. **JMeter Native Reporting**:
   * **View Results Tree**: For detailed response inspection.
   * **Summary Report**: Provides overall statistics like throughput, response time, and error rate.
   * **Graph Results**: Provides a graphical view of response times and request/response statistics.
2. **Enhanced Reporting**:
   * **JMeter Plugins**: Install the **JMeter Plugins Manager** and use plugins like **JMeter Dashboard Report**, which generates HTML-based reports with detailed charts, response time distribution, and other key performance metrics.
3. **Integrating with CI Tools**:
   * Use Jenkins to trigger JMeter tests and generate reports after every build.
   * Integrate **Allure** or **ExtentReports** to generate customized, interactive reports.
4. **Custom Report Generation**:
   * Use **JMeter’s JMX files** to configure **Post-Processors** that can process results and generate specific reports (e.g., extracting the average response time of certain requests).

These scenario-based interview questions and answers cover a broad range of topics related to **performance testing using JMX**. They touch on test framework setup, execution, troubleshooting, reporting, and optimization techniques, all crucial for working on performance testing projects.

Framework Interview Questions Scenario Based – Cucumber BDD

**1. Framework Design and Setup**

**Q1:** *You are tasked with setting up a Cucumber-based framework using Selenium for an e-commerce website. How would you structure the project and integrate Cucumber with Selenium?*

**Answer:**  
To set up a **Cucumber-Selenium** framework, I would follow these steps:

1. **Project Structure:**
   * **src/main/java**: Contains Java classes such as step definitions, page objects, and utility functions.
   * **src/test/java**: Contains test runners and Cucumber step definition classes.
   * **src/test/resources**: Contains feature files and any required configuration files (e.g., cucumber.properties).
   * **pom.xml**: Manages dependencies (e.g., Cucumber, Selenium, JUnit/TestNG, and other libraries).
2. **Dependencies in pom.xml:**
   * **Cucumber** for BDD syntax.
   * **Selenium** for web interactions.
   * **JUnit/TestNG** for test execution.
   * **WebDriverManager** to manage browser drivers.
3. **Cucumber Configuration:**
   * **Feature files** in src/test/resources to describe application behavior (e.g., login, add product to cart).
   * **Step Definitions** in src/test/java that implement the steps written in the feature file.
   * **Page Object Model (POM)** for abstraction of web elements in Selenium.
4. **Test Runner Class:**
   * A **JUnit** or **TestNG** runner to run the tests, integrating with Cucumber.
   * **CucumberOptions** annotation to define the location of feature files and step definition classes.

**Example Project Structure:**

plaintext

Copy

src/

├─ main/

│ ├─ java/

│ │ └─ pageobjects/ # Page Object Model classes

│ │ └─ utils/ # Utility functions (e.g., WebDriverManager)

└─ test/

├─ java/

│ └─ stepdefinitions/ # Step definition classes

│ └─ runners/ # Test runner classes (JUnit or TestNG)

├─ resources/

│ └─ features/ # Cucumber feature files

└─ pom.xml # Maven dependencies and configurations

**2. Test Case Writing and Step Definitions**

**Q2:** *You need to automate the login functionality for an e-commerce site. Write a Cucumber feature file to describe this scenario and the corresponding step definitions in Java using Selenium.*

**Answer:**  
**Feature File: login.feature** (located in src/test/resources):

gherkin

Copy

Feature: Login functionality

Scenario: Successful login with valid credentials

Given I navigate to the login page

When I enter valid credentials "user1" and "password123"

And I click on the login button

Then I should be redirected to the homepage

Scenario: Unsuccessful login with invalid credentials

Given I navigate to the login page

When I enter invalid credentials "user1" and "wrongpassword"

And I click on the login button

Then I should see an error message "Invalid login credentials"

**Step Definitions in Java** (located in src/test/java/stepdefinitions):

java

Copy

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import io.cucumber.java.en.\*;

public class LoginSteps {

WebDriver driver = DriverFactory.getDriver();

@Given("I navigate to the login page")

public void iNavigateToLoginPage() {

driver.get("https://www.ecommerce.com/login");

}

@When("I enter valid credentials {string} and {string}")

public void iEnterCredentials(String username, String password) {

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

usernameField.sendKeys(username);

passwordField.sendKeys(password);

}

@When("I click on the login button")

public void iClickLoginButton() {

WebElement loginButton = driver.findElement(By.id("loginButton"));

loginButton.click();

}

@Then("I should be redirected to the homepage")

public void iShouldBeRedirectedToHomePage() {

String currentUrl = driver.getCurrentUrl();

assert(currentUrl.contains("home"));

}

@Then("I should see an error message {string}")

public void iShouldSeeErrorMessage(String errorMessage) {

WebElement errorElement = driver.findElement(By.id("errorMessage"));

assert(errorElement.getText().equals(errorMessage));

}

}

**3. Running and Configuring Cucumber Tests**

**Q3:** *How would you configure the test execution for a Cucumber-Selenium project using* ***JUnit****? What configuration would you use in the* ***Test Runner*** *class?*

**Answer:**  
To configure and run Cucumber tests using **JUnit**, I would set up the following:

1. **Test Runner Class** (located in src/test/java/runners):

java

Copy

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features = "src/test/resources/features", // Path to the feature files

glue = "stepdefinitions", // Path to step definition packages

plugin = {"pretty", "html:target/cucumber-report.html", "json:target/cucumber.json"}, // Report formats

tags = "@smokeTest" // Optionally, run specific tagged scenarios

)

public class TestRunner {

// This class will be used to run the Cucumber tests using JUnit

}

1. **Explanation of @CucumberOptions:**
   * **features**: Path to the .feature files.
   * **glue**: Package containing step definitions.
   * **plugin**: Specifies report formats like **HTML** and **JSON** for the execution.
   * **tags**: Allows filtering based on tags like @smokeTest, @regression, etc.
2. **Running the tests**:
   * The tests are executed through the TestRunner class, and the results will be displayed as an HTML or JSON report after execution.

**4. Handling Data and Dynamic Content**

**Q4:** *How do you handle scenarios in Cucumber with dynamic data (e.g., user inputs, dynamic dropdowns)?*

**Answer:**  
To handle dynamic data and content in Cucumber, the following approaches can be used:

1. **Parameterization with Examples Table:**
   * Cucumber allows dynamic data using **Examples** in the **Scenario Outline**.

Example:

gherkin

Copy

Scenario Outline: Login with valid credentials

Given I navigate to the login page

When I enter valid credentials "<username>" and "<password>"

And I click on the login button

Then I should be redirected to the homepage

Examples:

| username | password |

| user1 | pass1 |

| user2 | pass2 |

1. **Using External Data Sources (CSV, Excel, JSON)**:
   * If data needs to be loaded from an external source, use **DataTables** to pass dynamic values from CSV, Excel, or other files.
   * **Example in Java**:

java

Copy

@When("I enter valid credentials from a CSV file")

public void iEnterCredentialsFromCSV() {

// Read from a CSV or Excel file using libraries like Apache POI or OpenCSV

}

1. **Dynamic Content Handling**:
   * If the page contains dynamic dropdowns or elements, use **explicit waits** (e.g., WebDriverWait) to wait until the elements are available or interactable.
   * Use **Page Object Model (POM)** to abstract web elements and their dynamic behaviors.

**5. Debugging and Troubleshooting**

**Q5:** *While running your Cucumber tests, you notice that some tests are failing intermittently due to synchronization issues. How would you address this problem in your framework?*

**Answer:**  
To resolve synchronization issues in Selenium-Cucumber tests:

1. **Explicit Waits**:
   * Replace Thread.sleep() with **explicit waits** using WebDriverWait to ensure elements are present before interacting with them.

java

Copy

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementId")));

1. **Fluent Waits**:
   * Use **FluentWait** for more customized wait conditions where we can define frequency, timeout, and exceptions to ignore.

java

Copy

FluentWait<WebDriver> wait = new FluentWait<>(driver)

.withTimeout(Duration.ofSeconds(10))

.pollingEvery(Duration.ofSeconds(1))

.ignoring(NoSuchElementException.class);

WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementId")));

1. **Handling Stale Element Reference**:
   * If you are getting StaleElementReferenceException, it may be because the DOM is refreshing. Re-find the element after such occurrences.
2. **Cucumber Tags for Flaky Tests**:
   * Use Cucumber **tags** to isolate problematic scenarios and rerun them individually for debugging.

**6. Advanced Topics**

**Q6:** *How do you integrate* ***Cucumber*** *with* ***CI/CD pipelines*** *to run tests automatically after each code push or deployment?*

**Answer:**  
To integrate **Cucumber** with **CI/CD** pipelines:

1. **Jenkins Integration**:
   * Add a Jenkins job to trigger **Cucumber** tests after each code push or deployment.
   * Use a **Maven**/**Gradle** job to run the tests, and configure the **Test Runner** class to execute the tests.
2. **Maven/Gradle Configuration**:
   * Ensure the pom.xml or build.gradle has the necessary **Cucumber**, **Selenium**, and **JUnit/TestNG** dependencies.
   * Use **Maven Surefire Plugin** or **TestNG Plugin** to run tests in the CI pipeline.
3. **Test Reporting**:
   * Generate and collect test reports (HTML, JSON) using **Cucumber Plugins** in Jenkins, which provide detailed feedback on test success or failure.
4. **Slack Notifications**:
   * Configure Jenkins or GitHub Actions to send notifications via **Slack** or **Email** on test results to keep the team informed.

Here are some **real-time XPath finding scenario-based interview questions** related to **Selenium**:

**1. Locating Elements on Dynamic Pages**

**Q1:** *You are automating a scenario where you need to select a product from a dynamically generated list of products. The product names and prices change with every page refresh. How would you write an XPath to select a product with a specific name from the list?*

**Answer:**  
In this scenario, the XPath needs to be able to identify elements dynamically, based on the text content. Here’s how you can do it:

**XPath:**

xpath

Copy

//div[@class='product']//h3[contains(text(), 'ProductName')]

**Explanation:**

* **//div[@class='product']**: This finds the div element that contains the product.
* **//h3[contains(text(), 'ProductName')]**: Finds the product name within an h3 tag, with partial matching based on the product name. Replace 'ProductName' with the actual product name or a variable in your script.

**2. Working with Tables**

**Q2:** *You need to select a row in a table where the first column contains the value "John Doe" and the second column contains the value "Manager". How would you find the row using XPath?*

**Answer:** In this case, the XPath should match both the first and second column values.

**XPath:**

xpath

Copy

//table[@id='employeeTable']//tr[td[1][contains(text(), 'John Doe')] and td[2][contains(text(), 'Manager')]]

**Explanation:**

* **//table[@id='employeeTable']**: Locates the table by its id.
* **//tr**: Selects the rows.
* **td[1] and td[2]**: Refers to the first and second columns of the table row.
* **contains(text(), 'John Doe')**: Matches the text in the first column.
* **contains(text(), 'Manager')**: Matches the text in the second column.

**3. Handling Hidden Elements**

**Q3:** *You need to interact with a hidden element that only becomes visible after hovering over another element. How would you find the hidden element using XPath once it becomes visible?*

**Answer:**  
When an element is dynamically shown after interacting with another element, you need to wait until it's visible. You can use contains() or @style attributes to identify such elements. Assuming the element becomes visible after hovering over a button:

**XPath to find hidden element:**

xpath

Copy

//div[@class='dropdown-menu' and contains(@style, 'display: block')]

**Explanation:**

* **@style**: Used to check if the display attribute is set to block (which makes the element visible).
* **contains(@style, 'display: block')**: Matches the style attribute when the element is visible.

**4. Working with Multiple Elements**

**Q4:** *You are automating a scenario where there are multiple buttons with the same class name, and you need to click on the second button. How would you find and click the second button using XPath?*

**Answer:** To select a specific button in a list of elements with the same class name, you can use indexing in XPath.

**XPath:**

xpath

Copy

(//button[@class='btn-primary'])[2]

**Explanation:**

* **(//button[@class='btn-primary'])**: Selects all buttons with the class name btn-primary.
* **[2]**: The index 2 selects the second button in the list.

**5. Working with Dynamic Attributes**

**Q5:** *You need to find a button that has a dynamically changing attribute value, such as a button with an ID that changes every time the page reloads but always contains the string "submit". How would you write an XPath for this?*

**Answer:** In such cases, you can use **contains()** to match a part of the dynamic value.

**XPath:**

xpath

Copy

//button[contains(@id, 'submit')]

**Explanation:**

* **contains(@id, 'submit')**: This matches the button whose id contains the substring submit, regardless of the dynamic parts of the id.

**6. Finding Elements with Specific Attributes**

**Q6:** *You are automating a scenario where you need to click on a link that has a specific data attribute. The data-id attribute changes dynamically for different links, but you need to find a link where the data-id equals "5". How would you write an XPath for this?*

**Answer:** You can use the exact match in XPath to select the link with the specific data-id value.

**XPath:**

xpath

Copy

//a[@data-id='5']

**Explanation:**

* **@data-id='5'**: Matches the link element where the data-id attribute equals 5.

**7. Handling Nested Elements**

**Q7:** *You need to click a button inside a div element with a specific class name, but the button is nested inside multiple divs. How would you construct an XPath to click the button inside the div with class product-details?*

**Answer:**

**XPath:**

xpath

Copy

//div[@class='product-details']//button[@class='buy-now']

**Explanation:**

* **//div[@class='product-details']**: Selects the div with the class product-details.
* **//button[@class='buy-now']**: Finds the button within that div with the class buy-now.

**8. Handling Input Fields**

**Q8:** *You need to fill out a form with fields that are dynamically generated. How would you select an input field with a placeholder "Enter your email"?*

**Answer:** You can use the placeholder attribute to identify the input field.

**XPath:**

xpath

Copy

//input[@placeholder='Enter your email']

**Explanation:**

* **@placeholder='Enter your email'**: Selects the input element whose placeholder attribute matches the given string.

**9. Selecting Radio Buttons or Checkboxes**

**Q9:** *You need to select a radio button with the label "Male" from a group of radio buttons. How would you locate and click the radio button?*

**Answer:** Radio buttons and checkboxes are often associated with labels, so you can use the label text to find the corresponding input element.

**XPath:**

xpath

Copy

//label[text()='Male']/preceding-sibling::input[@type='radio']

**Explanation:**

* **//label[text()='Male']**: Finds the label with the text "Male".
* **preceding-sibling::input[@type='radio']**: Selects the radio button input element that comes before the label.

**10. Advanced XPath - Parent and Child Relationships**

**Q10:** *You need to click on a button that is a child of a div element with a class name "container" and contains the text "Submit". How would you write the XPath?*

**Answer:**

**XPath:**

xpath

Copy

//div[@class='container']//button[text()='Submit']

**Explanation:**

* **//div[@class='container']**: Selects the div with class container.
* **//button[text()='Submit']**: Finds the button within that div that has the exact text "Submit".

**11. Selecting Based on Position**

**Q11:** *You need to click the last button on a page with multiple buttons. How would you write the XPath to select the last button?*

**Answer:**

**XPath:**

xpath

Copy

(//button)[last()]

**Explanation:**

* **(//button)**: Selects all the button elements.
* **last()**: Returns the last button in the list.

I worked on Jenkins Integration of Selenium Java Framework and running in master slave configuration, give me scenario-based Interview Questions on that

**1. Setting Up Jenkins with Selenium and Java Framework**

**Q1:** *You are tasked with setting up Jenkins for your Selenium Java automation framework. What are the key steps you would follow to integrate Selenium with Jenkins, and how would you ensure that the tests run on a slave machine instead of the master machine?*

**Answer:**

* **Install Jenkins on the master machine** and configure the **slave node** (a separate machine or VM) to run tests.
* **Install necessary plugins** on Jenkins like **Maven Integration**, **JUnit**, **Selenium**, and **SSH plugin** (if the slave is a different machine).
* On the **slave machine**, ensure **Java** and **Maven** are installed and properly configured.
* **Configure Jenkins Slave**:
  + Go to Jenkins > Manage Jenkins > Manage Nodes > New Node.
  + Add the slave machine with a name and configure it to use the appropriate **Java/Maven setup** for running the Selenium tests.
* **Set up a Jenkins Job** to run the Selenium tests:
  + **In the job configuration**, select the **Slave** node in the "Restrict where this project can be run" option.
  + Ensure **Maven** and **JUnit** plugins are used to compile and run the tests.
* **Configure build triggers** (e.g., GitHub webhook, scheduled build, etc.) to initiate the automation tests.

**2. Handling Parallel Execution on Master-Slave Configuration**

**Q2:** *How would you set up Jenkins to execute Selenium tests in parallel across multiple slave nodes to speed up the execution time of your automation tests?*

**Answer:**

* To execute tests in parallel across different slaves:
  + **Parallel Execution with Maven Surefire Plugin**: Update the pom.xml file to configure the Maven Surefire Plugin to run tests in parallel.

xml

Copy

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

<configuration>

<parallel>methods</parallel> <!-- Parallel method execution -->

<useUnlimitedThreads>true</useUnlimitedThreads> <!-- Unlimited threads -->

</configuration>

</plugin>

* + **Jenkins Setup**:
    - Use the **Matrix Build Plugin** or **Parameterized Builds** to trigger different tests on different slave nodes.
    - Configure Jenkins to assign different **test suites** to specific slave nodes (e.g., TestSuite1 to Slave1, TestSuite2 to Slave2).
    - You can also use the **"Run Tests in Parallel"** option in Jenkins if using tools like **TestNG** to run tests in parallel across different nodes.

**3. Handling Test Failures in Slave Nodes**

**Q3:** *You notice that the tests are failing only on the slave nodes, not on the master node. What steps would you take to troubleshoot and resolve this issue?*

**Answer:**

* **Check the Node Configuration**: Ensure that the slave node is configured with the correct **Java version**, **Maven**, and **environment variables** needed to run the Selenium tests.
* **Review Slave Node Logs**: Check the logs on the slave node for any errors related to dependencies, permissions, or environment setup issues.
* **Check for Dependency Mismatch**: Ensure that the same versions of **Selenium**, **WebDriver**, and other dependencies are available on both the master and slave nodes.
* **File Permissions**: Ensure that the files (like test scripts, logs, etc.) are accessible on the slave node and that the user running Jenkins has the necessary permissions.
* **Check Network Configuration**: Verify that the slave node has the required network access to external resources (e.g., browsers, web applications) that might be necessary for test execution.

**4. Jenkins Slave Node Availability**

**Q4:** *How would you handle the scenario when the slave nodes go offline and the Jenkins jobs fail to execute?*

**Answer:**

* **Automatic Node Restart**: Ensure that the **slave node** is configured to restart automatically if it goes offline. This can be done by configuring the slave node's connection as **"Launch agent via SSH"** with retry options.
* **Notifications**: Set up **Jenkins email notifications** to inform the team if any slave node goes offline.
* **Backup Nodes**: Set up a **backup slave node** or a **temporary slave node** that can be used when the primary slave goes offline.
* **Check Slave Logs**: Investigate why the slave node went offline by looking at the Jenkins system logs and the slave node logs.
* **Slave Node Monitoring**: Implement monitoring tools (e.g., Nagios, Zabbix) to detect when slave nodes are down and alert you in real-time.

**5. Handling Browser Drivers on Slave Nodes**

**Q5:** *Your tests are running on the slave nodes, but the browser isn't launching. What could be the issue, and how would you fix it?*

**Answer:**

* **Ensure WebDriver Setup**: Ensure that the required **browser driver (e.g., ChromeDriver, GeckoDriver)** is installed on the slave node and that it is available in the system **PATH** or specified in the test configuration.
* **Driver Compatibility**: Ensure that the versions of the **WebDriver** and the **browser** are compatible.
* **Check Permissions**: Make sure the Jenkins slave process has the correct permissions to execute the browser drivers on the slave machine.
* **Configure RemoteWebDriver**: If using **Selenium Grid** for parallel execution, make sure the **hub** and **node** are correctly configured, and that the **RemoteWebDriver** is pointing to the right slave node.
* **Headless Browsing**: For remote nodes without a graphical environment, ensure that **headless mode** is enabled for browsers like Chrome or Firefox.

**6. Optimizing Test Execution with Jenkins and Selenium**

**Q6:** *How would you optimize the test execution time when running Selenium tests on Jenkins with multiple slave nodes?*

**Answer:**

* **Parallel Test Execution**: As mentioned earlier, use the **Maven Surefire Plugin** or **TestNG Parallel Execution** to run tests in parallel on multiple slave nodes.
* **Optimize Test Suites**: Split the tests into smaller, independent suites to run on different slaves. For example, run regression tests, smoke tests, and sanity tests on different nodes.
* **Use Test Data Partitioning**: Use different sets of test data on each slave node to run subsets of the tests concurrently.
* **Reduce the Time for Browser Initialization**: Use **WebDriverManager** to manage and download the correct browser driver versions dynamically.
* **Use Caching**: Cache dependencies like Maven dependencies or browser drivers on the slave nodes to reduce the setup time.

**7. Configuring Jenkins Pipeline for Selenium Tests**

**Q7:** *You need to implement a Jenkins pipeline to execute your Selenium Java tests. How would you set up a Jenkins pipeline that includes stages like building, testing, and reporting in a master-slave configuration?*

**Answer:**

* **Define Pipeline in Jenkinsfile**: Create a Jenkinsfile to define the pipeline stages.
* **Use Declarative Pipeline Syntax**:

groovy

Copy

pipeline {

agent none

stages {

stage('Build') {

agent { label 'master' }

steps {

script {

sh 'mvn clean install'

}

}

}

stage('Test') {

agent { label 'slave-node' } // Use slave node for test execution

steps {

script {

sh 'mvn test'

}

}

}

stage('Report') {

agent { label 'master' }

steps {

script {

sh 'mvn surefire-report:report'

}

}

}

}

}

* **Explanation**:
  + **Agent**: Each stage can be configured to run on specific agents (master or slave nodes).
  + **Build Stage**: Runs on the master node to compile the code.
  + **Test Stage**: Runs on the slave node to execute Selenium tests.
  + **Report Stage**: Generates the test reports back on the master node.

**8. Scaling Jenkins and Selenium Grid**

**Q8:** *How would you scale Jenkins to handle a large number of test cases across multiple slave nodes?*

**Answer:**

* **Add More Slave Nodes**: Scale horizontally by adding more slave nodes to Jenkins. Each node can run a set of tests in parallel.
* **Use Selenium Grid**: Integrate **Selenium Grid** with Jenkins to distribute test execution across multiple machines.
* **Use Docker Containers**: Utilize **Docker** containers for easy setup of isolated environments for running tests. Each slave node could run in a separate container with its own set of browsers.
* **Use Cloud Solutions**: If scaling on-premises is a challenge, use cloud solutions like **AWS EC2**, **Azure**, or **Google Cloud** to dynamically provision slave nodes as needed.

**9. Integrating Test Reports into Jenkins**

**Q9:** *How would you integrate the test reports into Jenkins so that results from Selenium tests are easily visible after a build?*

**Answer:**

* **JUnit/TestNG Reports**: If using **TestNG** or **JUnit**, ensure that you generate test reports after each test run. Jenkins can automatically display these reports using the **JUnit plugin** or **TestNG plugin**.
* **Publish Test Reports in Pipeline**:
  + Use **JUnit plugin** in your pipeline to capture test results.

groovy

Copy

post {

always {

junit '\*\*/target/test-classes/testng-results.xml'

}

}

* + This will display the test results in Jenkins after every test run.

**COGNIZANT-INTERVIEW**

Programming:  
1- Program to Remove duplicates from Array without changing the order?  
input:  
int [] arr = {5,6,7,3,3,7,5};  
output:  
arr = {5,6,7,3};  
  
Java  
1-String str= "Akash"; can we change its value? if not then why? other classes to change it?  
2- Difference between String Buffer and String Builder classes?  
3-Diffrence between Arraylist and Linkedlist? which collection we will use for faster retrieval?  
4-Diffrence between Hashmap and Hashtable?  
5-Diffrence between == and equals to operator?  
  
Selenium:  
1- Write pseudo code for a scenario there is a dropdown on webpage. We need to select all values from and check its sorted or not?  
2- Explain the framework you have worked upon in current and previous projects?  
3- How to rerun the failed testcases without using TestNg.falied.xml file?  
4- Tell the order of Execution of testcases using TestNg framework with priority -2, -1 , 0, 1, 2?  
5- What is the order of Execution if no priority is set in TestNg alphabetical or ASCII value?  
6- How you are storing testdata and fetching into testcaes?tell me other ways apart from using excel file?  
7- Have you used cucumber? what is the use of Background keyword in Feature file?  
8- Difference between Scenario and Scenario outline?  
9- What are different keywords we use testrunner file using Junit? why we use Dryrun keyword?  
10-What is the use of Data providers in TestNg? write the psuedo code?  
11-How we run failed testcases using Listener class? tell me the implementation?  
  
Git:  
1-Tell me all the Git commands you are using in the project?  
2-How you will resolve the merge conflicts without manual check?  
  
Jenkins:  
How you are running the Script using Jenkins? how you will configure and nightly build?  
How many testcases you are running per day for regression suite?  
  
Manual:  
Tell me different Agile Ceremonies?  
How you are raising the defects? what is defect life cycle?  
What innovation you have done in your project? How you will integrate Gen AI in your testing?  
  
SQL:  
Write a query to find the Students id who has not enrolled to any courses?  
### Student Table  
| student\_id     | Student Name |  
| 1        | Akash     |   
| 2          | Vikash    |   
| 3        | Ashay |   
| 4       | Mohit    |   
  
### Courses Table  
| Course Name    | Course ID    | Student ID      |  
|--------------|----------------|----------------------|  
| Maths     | 110      | 1 |  
| Physics    | 111      |4 |  
| Science     | 112       | 1            |  
| English       | 113          | 4                |  
|-------------- |--------------|---------------------- |

**SCENARIO BASED-INTERVIEW**

Scenario Based Interview Questions for Automation Tester [**hashtag#QA**](https://www.linkedin.com/feed/hashtag/?keywords=qa&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7307649474703540225)  
  
🔹 Your automated tests for a login page are failing intermittently. How would you investigate and address this issue?  
  
🔹 The website you're testing has elements that constantly change IDs or positions. What techniques would you use to create reliable locators for your tests?  
  
🔹 A critical API endpoint is returning unexpected errors. How would you use automation to help debug this?  
  
🔹 Your web application needs to work flawlessly across different browsers (Chrome, Firefox, Safari). How would you approach cross-browser test automation?  
  
🔹 Users complain about slow loading times on a specific page. How can automation help identify performance issues?  
  
🔹 You need to test a form with a large number of different input values. How would you efficiently manage and use test data in your automation?  
  
🔹 The company is prioritizing mobile users. How would you automate testing on different mobile devices and operating systems?  
  
🔹 You encounter a feature that seems impossible to automate (e.g., audio/video playback). How do you handle this in your test strategy?  
  
🔹 You're tasked with automating tests for an old application with limited documentation. How would you approach this?  
  
🔹 How would you integrate your automated tests into a Continuous Integration/Continuous Delivery pipeline?

[**#AUTOMATION**](https://www.linkedin.com/feed/hashtag/?keywords=automation&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7308429516513714177) **TESTING INTERVIEW QUESTIONS:**  
  
1. Explain the automation framework.  
2. What unit testing framework are you using?  
3. Difference between JUnit and TestNG.  
4. How can you rerun failed test cases?  
5. Which CI/CD tool have you used?  
6. Do you know how to create a parameterized pipeline job?  
7. Write a program to validate usernames (alphanumeric, no spaces/special characters, no duplicates).  
8. Write a program to count character occurrences in a string.  
9. Find the highest number in an array without using inbuilt methods.  
10. Reverse a string without using inbuilt methods.  
11. Difference between String and StringBuffer.  
12. Difference between private, public, and protected access modifiers.  
13. Explain the collections framework hierarchy.  
14. Difference between Hashtable and HashSet.  
15. Difference between Comparable and Comparator.  
16. Explain the exception hierarchy.  
17. Write a program to create a custom exception.  
18. What is an interface and an abstract class?  
19. Why is an abstract class allowed to have a constructor?  
20. Difference between final, finally, and finalize.  
21. Explain polymorphism.  
22. Write a program to override a function.  
23. Do you know about Java 8?  
24. What are Java Streams and Lambda Expressions?  
25. Difference between git rebase and git merge.  
26. When to use git rebase?  
27. What is a functional interface?  
28. Difference between status codes 401 and 403.  
29. Difference between OAuth 1.0 and OAuth 2.0.  
30. Write a program using the RestAssured library with assertions for GET/POST calls.  
31. What is the content type in OAuth 2.0?  
32. Write a SQL query using JOIN (two tables with columns given).  
33. Handle multiple windows in Selenium.  
34. Capture screenshots in Selenium.  
35. How do you handle synchronization in automation scripts?  
36. Write the syntax for Fluent Wait.  
37. Read/write data from an Excel file.  
38. Different stages of an automation test.  
39. Types of testing.  
40. Difference between priority and severity.  
41. Fields required to log a bug/defect.  
42. What is a test plan and what does it include?  
43. Briefly explain the phases of SDLC.

**I'M SHARING A LIST OF SDET INTERVIEW QUESTIONS PROVIDED BY ONE OF MY MENTEES FOR** [**HASHTAG#PUBLICISSAPIENT**](https://www.linkedin.com/feed/hashtag/?keywords=publicissapient&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7309207619158581249) **| 4+ YEARS EXPERIENCE**  
  
[**hashtag#Round1**](https://www.linkedin.com/feed/hashtag/?keywords=round1&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7309207619158581249) : (Virtual) - Duration: 1 hr   
1 Explain Object Oriented Programming(OOPS) concepts in depth  
2 Two core Java output based questions   
a static keyword used  
b ++operator based  
3 In a give string found the count of each character and replace it whenever its duplicate Input="Automation", Output="2u22m22i2n"  
4 Write all the git commands you have used in the project  
5 Have you configure Jenkins pipeline and how to trigger it daily at night at particular time   
6 What is String constant pool ? Why String is immutable   
7 What is serialization and deserialization concept  
8 Different types of inheritance in Java ? What is multilevel inheritance  
9 TestNG annotations  
10 What are listeners in TestNG ? How you have used in the script  
11 Which framework you have used and how you are using testdata in the script  
12 Difference between findElement and findElements in Selenium  
13 What is return type of getWindowhandles method in selenium  
  
[**hashtag#Round2**](https://www.linkedin.com/feed/hashtag/?keywords=round2&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7309207619158581249) :(Face To Face), Duration 1hr  
1 Given one scenario to find the text from XPath and write code for it  
Open Hdfc bank website -> mousehover Login Button -> Click on Login Button-> Click on Know more -> Capture all the language text   
2 How you are validating JSON schema in the project ? Write down code for it  
3 Given one JSON and asked to create the post request and write code to fetch all the cities for which temp is greater than 40  
4 What are different authentication schemas ? Difference between 401 and 403 authentication request  
5 What are selenium 4 features and selenium webdriver architecture  
6 Difference between Rest and Soap and Rest Web Service architecture  
7 How we do execution using Selenium grid ? Why do we use Selenium grid   
8 What are listeners in TestNG and how you are implementing it  
9 Why we use data providers in TestNG ? Write down the code  
10 How we do parallel execution in TestNG ? Why we pass test name in TestNG.xml  
11 How selenium picks cucumber file ? Syntax of Testrunner file ? What does dryrun = true does  
12 What is Test grooming ? What are agile ceremonies  
13 Have you worked any cloud technology in your project like AWS, Azure  
  
[**hashtag#Round3**](https://www.linkedin.com/feed/hashtag/?keywords=round3&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7309207619158581249) : HR Discussion   
  
Struggling to crack that interview? Join 6-week intensive program designed to fast-track your success!  
  
--> Master coding & automation with hands-on practice  
--> Tackle real-world scenarios with expert guidance  
--> Build confidence with proven interview strategies  
--> 6 mock interviews to ensure you're interview-ready  
  
Drop me a message on LinkedIn if you're interested!

[**#SCENARIO**](https://www.linkedin.com/feed/hashtag/?keywords=scenario&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7309416389750988800) **TEST AUTOMATION INTERVIEW QUESTIONS**  
  
(1) You are asked to automate a functionality that is not yet fully developed. How do you handle this?  
(2) If a Test fails, what will be your next step?  
(3) If the application has minor changes, what would be your approach to modifying the Automation scripts?  
(4) How would you automate login functionality for a website?  
(5) How would you automate a Test scenario where you need to check if an email is sent after a user registration?  
(6) If there is a scenario that takes a long time to execute, would you prefer Manual TestNG or Automation Testing? Why?  
(7) How would you automate a scenario where you need to validate the contents of a downloaded file after clicking a button on a webpage?  
(8) How would you automate a scenario where you need to verify a specific color, font, and position of an element on a webpage?  
(9) How would you handle pop-up windows or alert boxes in your Automation script?  
(10) How would you automate a scenario where you need to verify if a user is able to scroll down a webpage until the footer section is visible?  
(11) You've been asked to automate a legacy application. What is your approach?  
(12) A script you wrote was working fine yesterday but is failing today. How do you troubleshoot?  
(13) Your Automation scripts are running slowly. How can you improve the speed?  
(14) Your Automation script is failing due to a change in the application. How do you handle this?  
(15) How would you automate a scenario where you need to verify a specific color, font, and position of an element on a webpage?

**EY - Interview Questions for QA Automation/SDET role (2-5 years)**  
  
1. Write a Java Program for given Strings are anagram or not str1 = "tomato", str2 = "matoto"?  
2. Write a Java program to remove duplicates from given String str1 = "automation": o/p: "automin"?  
3. What is the strategy to start failure analysis from Day 1 itself in current release ?  
4. How can you calculate ROI for automation?  
5. How can you assign the work when you have Senior and junior(1-2 Years) team members?  
6. as a lead QA, what is your checklist for code review?  
7. How to perform cross browser testing?  
8. How can you execute failed test cases without manual intervention. Here we can use retry mechanism?  
9. How to execute all failed test cases at once after initial execution?  
10. Architecture for Selenium 3 and Selenium 4  
11. What is topmost interface in Selenium?  
12. How do you pick test cases which are prone to defects?  
13. Can you determine the total execution time of tests with preparation of any dependencies like setup of mocks for payment gateways or test data preparation?  
14, How to identify the test cases that can be automated?

**QA Interview Questions with Answers:**  
  
1. You are asked to automate a functionality that is not yet fully developed. How do you handle this?  
  
Approach: I’d start by collaborating with the development team to understand the current state of the functionality, its expected behavior, and the timeline for completion. Since it’s not fully developed, I’d avoid writing full automation scripts immediately. Instead, I’d:  
Create a test plan or skeleton framework based on the requirements or design specs.  
Use stubs or mocks to simulate undeveloped parts (e.g., API responses or UI elements) if possible.  
Focus on automating stable components first, like login or navigation, that interact with the feature.  
Wait for a stable build before finalizing scripts, ensuring I don’t waste time on frequent rework.  
Key Point: Communication with developers and prioritizing flexibility in the script design are critical to adapt as the feature evolves.  
  
2. If a Test fails, what will be your next step?  
Approach: My immediate step is to investigate the failure:  
Reproduce: Rerun the test to confirm if it’s consistent or intermittent.  
Analyze Logs: Check test logs, screenshots, or videos (if captured) to pinpoint the issue.  
Verify Application: Manually test the scenario to see if it’s a product bug or a script issue.  
Debug Script: If it’s a script issue (e.g., locator change), update the script and retest.  
Report: If it’s a bug, log it with details (steps, environment, screenshots) in a tool like Jira and inform the team.  
Key Point: I’d determine whether the failure is due to the application, environment, or script, then act accordingly—fix the script or escalate the defect.  
  
3. If the application has minor changes, what would be your approach to modifying the Automation scripts?  
Approach: I’d assess the impact of the changes and update the scripts efficiently:  
Identify affected areas (e.g., UI elements, workflows) by reviewing change logs or developer input.  
Update locators (e.g., XPath, CSS) if UI elements like IDs or classes changed.  
Refactor code if logic or flow is altered, keeping it modular to minimize rework.  
Use parameterization or page object models (POM) to limit changes to specific files.  
Rerun tests to validate updates and ensure no new issues arise.  
Key Point: A well-designed, modular framework (like POM) reduces maintenance effort for minor changes.

[**#Deloitte**](https://www.linkedin.com/feed/hashtag/?keywords=deloitte&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7311959861414531072) **Senior Test Engineer Interview Questions ⁉️**  
  
##[**hashtag#Round**](https://www.linkedin.com/feed/hashtag/?keywords=round&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7311959861414531072) 1: (Technical)\*\*   
  
1. Explain about yourself and your roles and responsibilities.   
2. Write Java code to reverse a string while preserving the whitespace.   
3. Given an SQL table, find the second-largest salary of an employee.   
4. Explain the concept of joins and list the different types of joins available.   
5. What is LinkedHashMap? Explain its use.   
6. Explain black-box testing and white-box testing.   
7. What is the difference between an exception and an error?   
8. Difference between findElement and findElements.   
9. What is the difference between implicit wait and explicit wait?   
10. Write Selenium code to automate a calendar WebElement.   
11. How would you fetch the text from a text box in Selenium?   
12. How do you enter text in an alert using Selenium?   
13. What is the difference between checked and unchecked exceptions?   
14. If an XPath selects two elements on a webpage, does findElement throw an exception?   
15. Explain smoke testing and sanity testing.   
16. What are relative locators in Selenium?   
17. Explain the challenges you faced in your project.   
18. How would you pick test cases for regression testing?   
  
---  
  
##[**hashtag#Round**](https://www.linkedin.com/feed/hashtag/?keywords=round&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7311959861414531072) 2: (Techno-Managerial)\*\*   
1. Explain your framework in detail.   
2. In your framework, if you have 100 pages, do you create 100 Page Objects?   
3. What is an “Element click intercepted” exception, and how do you fix it?   
4. How do you take screenshots for failed test cases only in TestNG?   
5. Explain how to connect your test cases with Azure.   
6. What is the get fetch command?   
7. In Postman, how do you delete a variable after test execution is finished?   
8. Why do you prefer using Cucumber BDD in your project?   
9. Explain dynamic binding and static binding.   
10. What is method overloading, and what is method overriding?   
11. What is the difference between Comparable and Comparator?   
12. How would you conclude that a login page is user-friendly?   
13. What are 5 points you consider when writing a good test case?   
14. Suppose a developer is not fixing a bug; how would you approach the situation?   
15. What are threads in JMeter?   
16. Can you automate captchas?   
17. In what cases would you require fluent waits?   
18. In Selenium, how would you open a new tab?   
19. In Jenkins, what is the purpose of a CRON expression?   
  
#[**hashtag#Round**](https://www.linkedin.com/feed/hashtag/?keywords=round&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7311959861414531072) 3 ( Hr Discussion )

**API Testing Interview Questions from BIRLA-SOFT :**  
Round #1 :   
1) How would you configure to run parallel API tests in TestNG or JUnit?  
2) How can you parameterize API tests to run against different environments (dev, QA, prod) dynamically?  
3) How do you validate nested JSON responses when the structure is dynamic?  
4) How do you extract and validate specific fields from an array of JSON objects in a response?  
5) How would you handle API responses that return different data types dynamically for the same field?  
6) Given an API that returns paginated results, how would you validate data consistency across multiple pages?  
7) If an API rate limits requests (e.g., throttling at 100 requests per minute), how would you implement retries with exponential backoff ?  
8) How to handle APIs that randomly return 429 (Too Many Requests) errors?  
9) If an API triggers an asynchronous process (e.g., starts a batch job and returns a Job ID), how would you validate its completion?  
10) How do you handle APIs that return multipart/form-data responses?  
11) How to simulate 1000 concurrent users hitting an API?  
12) How would you integrate Rest Assured with JMeter or Gatling for performance testing?  
13) How can you automate testing for unauthorized access scenarios using Rest Assured?  
  
➡️ Coding part :   
  
14) Open a Notepad , Can you write a test that measures API response time and fails if it exceeds a threshold?

**VALUELABS INTERVIEW QUESTIONS : SDET 4.5 TO 6 YEARS 😊😊**  
  
Duration decided: 30 minutes  
  
Actual Interview: 48 minutes.  
  
1. Please introduce yourself.  
2. Please explain your Automation Framework, all the components.  
3. What is a Page Object Model?  
4. How do you run your test cases in parallel in Cucumber?  
5. Explain the contents of the Runner File in Cucumber?  
6. What is a Singleton Design Pattern?  
7. What are the advantages and disadvantages of the Page Object Model?  
8. What is Selenium Grid?  
9. Explain the WebDriver create statement line?  
11. Explain the Maven Lifecycle?  
12. How do you run the failed test cases?  
13. How do you generate Reports in Selenium?  
14. How do you customise reports after your test execution?  
15. What kind of waits are there in Selenium?  
16. Write the Code Snippet for Explicit Wait?  
17. Write the Code Snippet for Drag and Drop in Selenium?  
18. How do you switch to different Windows in Selenium?  
19. Why do we use SET in Window Handles?  
20. Write the Code for taking screenshot in Selenium?  
21. What is the difference between Scenario and Scenario Outline in Cucumber?  
22. How do you pass data to your Selenium Scripts?  
23. How do you decide the priorities of your Test Cases?  
24. If you want to execute one test case again and again how do you do that?  
25. What are the different annotations used in TestNG?  
26. Write the hierarchy of annotations in TestNG?  
27. What is the defect life cycle?  
28. What is the difference between Agile and Waterfall Model?  
29. What is the difference between 201 and 204 Status Code?  
30. What is the difference between 401 and 403 Status Code?  
31. What are the components of an API Request?  
32. What is the difference between Query Parameters and Path Parameters?  
33. How do you resolve Conflicts in Git?  
34. What is the difference between git pull and git patch?  
35. Explain the use of Jenkins in the Automation Framework?

[**#CGI**](https://www.linkedin.com/feed/hashtag/?keywords=cgi&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A7314542019752669185) **- SDET/QA AUTOMATION INTERVIEW QUESTIONS | 3-6 YEARS OF EXPERIENCE**  
  
1 Tell me something about yourself   
2 What is the final keyword in Java  
3 What is a constructor ? Can we override a constructor  
4 What is abstraction  
5 How to achieve multiple abstraction in Java  
6 What is an interface  
7 What is polymorphism ? Where have you applied it in your framework  
8 What is method overloading and method overriding ? Explain both and where you used them in your framework  
9 What is memory management in Java  
10 Difference between constructor overloading and method overloading  
11 What are exceptions in Java  
12 List some exceptions you faced during automation and how you handle them  
13 How do you handle dynamic web elements  
14 Write the syntax of the Select class and explain its methods  
15 How do you handle iFrames  
16 How do you handle mouse actions in Selenium  
17 How do you handle keyboard actions in Selenium  
18 How do you press the ENTER key using the Actions class ?Write the syntax  
19 How do you use the SHIFT key to type uppercase letters in Selenium? Write the syntax  
20 How to handle dynamic elements  
21 How to print all the links from a webpage ?Where do you store those links  
22 How to check for broken links ?How would you validate each link if there are 20 links in web page  
23 What is finally block in Java  
2 Difference between throw and throws  
24 Difference between findElement and findElements  
25 Explain your framework Structure? How do you perform parallel execution in your framework  
26 What are your cucumber options  
27 If you join a new project, what git commands do you use from start to end? What are the stages you follow  
28 Do you have any idea about SQL ? Tell me few queries you performed ?  
29 Do you have any idea Linux commands ?  
  
Scenario Based Question  
  
1 You have 3 frames(Frame A, Frame B, Frame C).After clicking Frame C, the application opens another tab containing a login, password, and sign-in function ? Write the Selenium automation script for this scenario ?

Mock Interview Script: Kiran Prakash Bellad

# Introduction

## Can you quickly introduce yourself and your experience?

Sure. I’m Kiran Prakash Bellad, a QA Engineer with over 4 years of experience in test automation, mainly working with Selenium, Postman, Rest Assured, and Java. I’ve developed automation frameworks from scratch using TestNG and Cucumber, and I’ve also done performance testing with JMeter. I'm currently working at Dassault Systems, where I focus on both UI and API automation, integrating our tests with CI/CD pipelines through Jenkins. I’m passionate about writing clean, reusable code and ensuring quality across agile releases.

# Technical

## What is the Page Object Model, and how have you used it?

Page Object Model is a design pattern where each web page is represented by a Java class with page elements and actions encapsulated. It improves code reusability and maintainability. In my current project, I used POM with Selenium and TestNG to modularize UI tests, which helped reduce redundancy and made it easier to update locators when the UI changed.

## How did you structure your API automation framework using Rest Assured?

I created a base class with reusable methods for setting up headers, sending requests, and parsing responses. I used POJOs for request/response payloads and managed test data using external JSON files. All validations were done with Hamcrest matchers, and the tests were tagged with BDD tags like @SMOKE and @REGRESSION for selective execution.

## Can you explain your use of Cucumber in your last project?

Yes, we used Cucumber for behavior-driven development. I wrote feature files with Gherkin syntax and implemented step definitions in Java. We used scenario outlines with Examples for data-driven tests, and organized reusable steps to maintain consistency. It helped bridge the gap between testers and product owners since they could read and validate our feature files.

## How have you integrated your tests with Jenkins?

In Jenkins, I created jobs that fetch the latest code from Git, compile it using Maven, and execute TestNG test suites. I parameterized jobs to select environments like QA or Staging. For reporting, I configured emailable HTML reports and integrated Slack notifications for build results. This made our testing process fast, repeatable, and visible to the team.

## What do you test with JMeter, and how?

I used JMeter to test API performance, simulating multiple concurrent users with thread groups. I measured metrics like response time, error percentage, and throughput. For example, we tested our registration API under load to ensure it could handle peak traffic. I also integrated JMeter results with Jenkins to trigger tests nightly.

# Qa Strategy

## How do you handle test case creation in Agile?

I write test cases directly from user stories during sprint planning. I prioritize critical paths first, then edge cases. I also maintain regression test suites and keep them updated every sprint. We collaborate with developers in daily standups to align on functionality and testing expectations.

# Communication

## Tell me about a challenging bug and how you solved it.

We had an intermittent login failure in the production environment. I analyzed logs, recreated the issue in lower environments, and found it was due to session caching inconsistencies. I worked with the dev team to modify session handling, added tests to catch it early, and confirmed the fix across browsers and environments.

## Why should we hire you?

I bring hands-on experience in test automation, deep understanding of QA processes, and the ability to build robust frameworks from scratch. I'm also proactive in learning and adapting to new tools. More importantly, I care deeply about product quality and I bring both technical skill and teamwork to deliver reliable software.

**1. Tell me about your experience with test automation frameworks.**

**Answer:**  
I have extensive experience in developing test automation frameworks from scratch. My main expertise lies in using Selenium with Java for UI automation. I’ve also built test automation frameworks utilizing Rest Assured for API testing and Cucumber for behavior-driven development (BDD). In my projects, I’ve focused on creating reusable and maintainable test scripts, handling dynamic elements, and integrating the frameworks with CI/CD tools like Jenkins to ensure smooth execution of tests in different environments.

**2. Can you explain how you handle dynamic elements in Selenium?**

**Answer:**  
Dynamic elements, such as those with changing IDs or class names, can be tricky. I use different strategies like:

* **Explicit waits** (WebDriverWait) for elements to appear or be clickable before performing actions.
* **XPath** strategies that are based on attributes like contains(), starts-with(), or text-based searches.
* Using **CSS selectors** with more reliable attributes like data-test IDs or custom attributes to identify elements.
* If an element’s ID or class is not stable, I try to locate it through its surrounding structure, utilizing relative XPath expressions.

**3. What is the role of Cucumber in your test automation strategy?**

**Answer:**  
Cucumber helps bridge the gap between technical and non-technical stakeholders by enabling Behavior-Driven Development (BDD). I use it to write feature files in Gherkin syntax, which are easy to understand for both developers and business analysts. These feature files define test scenarios that are then mapped to step definitions in Java. This allows for collaborative test case writing and ensures that the requirements are well-aligned with the test cases, making the process more transparent and easier to maintain.

**4. What’s your approach for testing REST APIs?**

**Answer:**  
For testing REST APIs, I use **Rest Assured** as it provides a rich set of features for sending HTTP requests, validating responses, and performing various assertions. My approach typically involves:

* Verifying HTTP response codes to ensure correctness (e.g., 200 for successful requests).
* Validating response payloads using JSON or XML schema validation.
* Checking response headers for correct content types or authentication tokens.
* Automating the process of testing various endpoints with different request methods (GET, POST, PUT, DELETE) and parameters.

**5. How do you integrate your test scripts with a CI/CD pipeline?**

**Answer:**  
I integrate my test automation scripts into the CI/CD pipeline using **Jenkins** or similar tools. This involves:

* Setting up Jenkins jobs to trigger tests on code check-ins or pull requests.
* Ensuring the tests are executed in different environments to simulate real-world scenarios.
* Configuring the Jenkins pipeline to run tests in parallel, improving execution speed.
* Sending reports after every test run using plugins like Allure or TestNG, making it easy to track test results and failures.
* Setting up notifications for developers to receive alerts on test failures, helping to resolve issues quickly.

**6. Can you walk me through your process for writing effective test cases?**

**Answer:**  
When writing test cases, I focus on:

* **Clarity**: Each test case must clearly define the objective and expected outcome.
* **Modularity**: I try to write reusable test steps to avoid redundancy.
* **Preconditions**: I ensure that the necessary setup (like test data or environment setup) is clearly mentioned.
* **Test Data**: I ensure the data used in test cases is varied, covering both valid and invalid scenarios.
* **Error Handling**: The tests are written to capture any unexpected system behavior, including error messages, and compare them against expected outputs.
* **Traceability**: I ensure that test cases are linked back to requirements or user stories, so the tests are aligned with project goals.

**7. What are the biggest challenges you face with Selenium and how do you overcome them?**

**Answer:**  
One of the main challenges with Selenium is handling **browser compatibility** issues, as different browsers may behave differently. To tackle this:

* I ensure that tests are written in a way that can easily be executed across multiple browsers using Selenium Grid.
* I regularly update WebDriver binaries and ensure compatibility with the browsers.
* Another challenge is **flaky tests**, which can happen due to synchronization issues or timing delays. I resolve this by using explicit waits to handle the timing of element visibility and interaction.
* Occasionally, I deal with **pop-ups** and **alerts**. I handle these by using WebDriver’s built-in Alert interface and switching to the alert when necessary.

**8. How do you ensure test reliability and prevent flaky tests?**

**Answer:**  
To ensure test reliability, I focus on the following:

* **Proper synchronization**: I use explicit waits to make sure elements are available before interacting with them.
* **Isolating tests**: Each test is independent and doesn’t rely on the state of previous tests, reducing dependencies and improving reliability.
* **Data cleanup**: After every test, I clean up any test data or restore the system to a known state.
* **Regular maintenance**: I regularly review and refactor test cases to keep them efficient and relevant, ensuring they reflect the current system behavior.

**9. How do you handle version control for your test scripts?**

**Answer:**  
I use **Git** for version control, keeping my test scripts in a repository alongside the main codebase. This allows:

* Easy collaboration with team members, where everyone can contribute to the automation scripts.
* Tracking changes made to test cases and scripts through Git commits.
* Keeping the scripts in sync with application updates, ensuring that the test cases are always up-to-date.
* Creating branches for different features or bug fixes and ensuring smooth integration with the main branch through pull requests and code reviews.

**10. How do you prioritize which tests to automate?**

**Answer:**  
When deciding which tests to automate, I prioritize:

* **Regression tests**: These tests are executed frequently and benefit from automation.
* **High-risk areas**: Features or components that are critical for the application’s success and may cause major issues if broken.
* **Tests with stable functionality**: I avoid automating tests for unstable features or areas that change frequently.
* **Time-consuming tests**: Long-running tests or manual processes that are repetitive can be automated to save time and increase efficiency.

### ****Top Screening Questions (HR or Tech Recruiter Round)****

**1. Can you give me a quick overview of your experience in test automation?**  
I have extensive experience in test automation, primarily working with Selenium, Java, and Rest Assured. I have developed and maintained test automation frameworks from scratch, focusing on both API and UI automation. I specialize in building reusable, modular frameworks using TestNG and Cucumber, ensuring that the automation suite supports both functional testing and continuous integration. I’ve worked in agile teams to automate regression and smoke tests and ensure rapid feedback during development cycles.

**2. What tools and technologies are you most comfortable with right now?**  
I am most comfortable with Selenium for UI automation, Java for writing test scripts, and Rest Assured for API testing. I also have experience using Cucumber for behavior-driven development (BDD) and TestNG for managing test execution. For CI/CD, I regularly use Jenkins, and I work with Git for version control. Additionally, I am proficient with tools like Postman for manual API testing and JMeter for performance testing.

**3. Tell me about your current role at Dassault Systems. What do you test there?**  
In my current role at Dassault Systems, I work as a Senior QA Engineer where I lead the development of automation frameworks for both web and API testing. I primarily test SaaS applications, focusing on verifying functionality, performance, and security. I create automated test cases for UI interactions using Selenium and API calls using Rest Assured, ensuring the robustness of the application across multiple platforms.

**4. Why did you transition from Mechanical Engineering to QA/IT?**  
I transitioned to QA/IT because of my growing interest in problem-solving, coding, and automation. While Mechanical Engineering was a solid foundation, I was drawn to the dynamic nature of software testing, where I could use my analytical skills to design and automate tests, ensuring high-quality software. The transition allowed me to tap into my passion for technology and to work on projects that continuously evolve and challenge me.

**5. Are you comfortable working on both API and UI automation? Give examples.**  
Yes, I am comfortable working on both API and UI automation. For UI automation, I have built frameworks using Selenium and TestNG, focusing on cross-browser testing and integrating with CI/CD pipelines. For API automation, I use Rest Assured to validate JSON responses, ensure correct status codes, and verify data consistency across endpoints. In my previous project, I automated both API and UI testing for a financial application to ensure end-to-end functionality.

**6. Which testing framework have you built or worked with from scratch?**  
I have built test automation frameworks from scratch for both UI and API testing. My UI framework uses Selenium with Java, organized in the Page Object Model (POM) pattern for easy maintenance. For API testing, I’ve built a Rest Assured-based framework that integrates with TestNG for reporting and supports data-driven testing. I’ve also integrated these frameworks with Jenkins for continuous test execution in the CI/CD pipeline.

**7. Can you explain how you used Jenkins in your projects?**  
I have used Jenkins to automate the execution of test suites in our CI/CD pipeline. My primary use of Jenkins involves triggering test execution on code commits or pull requests, running tests across multiple browsers, and reporting results back to the team. I’ve also used Jenkins to integrate with Docker containers to run tests in isolated environments, ensuring consistent results across different test environments.

**8. How do you collaborate with developers or product teams in an Agile environment?**  
In an Agile environment, I collaborate closely with developers and product teams during daily stand-ups and sprint planning sessions. I participate in discussions around feature development, providing feedback on test coverage early in the process. I also work with developers to write and maintain automation scripts for new features, ensuring that the testing is integrated into the development cycle. Additionally, I help product teams by translating requirements into clear, testable scenarios.

**9. Why are you looking for a new opportunity now?**  
I am looking for a new opportunity because I’m seeking new challenges that will allow me to grow further in the field of test automation and work with cutting-edge technologies. I want to contribute to a high-performing team where I can continue building robust automation frameworks while expanding my knowledge in advanced testing tools and methodologies.

**10. Are you open to relocation or remote work?**  
Yes, I am open to both relocation and remote work, depending on the needs of the role and the company. I believe that flexible working conditions enhance productivity and collaboration, so I’m comfortable adapting to either setting.

### ****Technical Questions (Based on Your Resume)****

**Selenium & UI Automation**

**1. What is the Page Object Model, and why did you use it?**  
The Page Object Model (POM) is a design pattern that encourages creating an object for each web page of the application. It helps to reduce code duplication and enhances test maintenance by keeping the test scripts separate from the page-specific details. I used POM to structure my Selenium test scripts for better readability, reusability, and maintainability, allowing the team to manage UI changes without breaking the test scripts.

**2. How did you handle dynamic elements or XPath issues in Selenium?**  
For dynamic elements, I handle issues by using relative XPath expressions based on stable attributes such as data-test-id, name, or class. I also use **Explicit Waits** to wait for elements to become visible or interactable before performing actions. If the element's location is highly dynamic, I use strategies like contains() or starts-with() in XPath to make the locators more flexible.

**3. How does TestNG help in cross-browser execution?**  
TestNG supports parallel execution of tests across different browsers, which is useful for cross-browser testing. By using TestNG’s @Parameters annotation, I can define different browsers (e.g., Chrome, Firefox, Edge) as parameters and configure the tests to run across those browsers. This allows us to ensure that the application functions correctly on different platforms without having to manually execute tests on each browser.

**4. How do you manage data-driven testing in Selenium?**  
I manage data-driven testing in Selenium by using TestNG’s data provider feature, which allows the execution of the same test with multiple sets of data. This approach ensures that I can test different scenarios and edge cases without writing duplicate code. I also use external data sources like Excel or CSV files to provide test data for scenarios like login, registration, and form submissions.

**API Testing (Postman & Rest Assured)**

**1. How do you structure a Postman collection for automation?**  
In Postman, I organize collections based on different services or endpoints. I create folders within collections for better organization and group tests based on functionality. For automation, I export collections to integrate them with CI/CD tools like Jenkins, where they can be triggered for automated execution. I also make use of environment variables to manage different environments like development, staging, and production.

**2. What kinds of assertions do you use in Postman scripts?**  
In Postman scripts, I use assertions to verify HTTP status codes, response time, headers, and response body content. For example, I assert that the response code is 200 OK, that the content type is JSON, and that certain fields in the JSON body match expected values. I use pm.response.to.have.status() and pm.response.to.have.jsonBody() for these checks.

**3. Explain how you automated REST APIs using Rest Assured.**  
I automated REST APIs using Rest Assured by sending HTTP requests (GET, POST, PUT, DELETE) to different endpoints and validating the response. I use assertions to check for expected status codes, response times, and the correctness of the returned data. I also integrate the Rest Assured framework with TestNG to organize the tests, and I use data-driven testing to cover multiple input scenarios for APIs.

**4. What is a POJO and how did you use it in your API framework?**  
A POJO (Plain Old Java Object) is a simple Java class with fields that represent data, typically used for deserialization of API responses. I use POJOs in my API framework to map JSON responses to Java objects. This allows for easy validation of the data by asserting values against expected results in the test cases.

**BDD with Cucumber**

**1. How do you write a Cucumber scenario for a login API?**  
I write a Cucumber scenario for a login API in Gherkin syntax, specifying the steps involved in logging in:

gherkin

CopyEdit

Feature: Login API

Scenario: Successful login with valid credentials

Given I have valid login credentials

When I send a POST request to the login endpoint

Then I should receive a 200 status code

And the response body should contain a token

This scenario is mapped to step definitions in Java, where I use Rest Assured to send the POST request and validate the response.

**2. What are hooks in Cucumber and how do you use them?**  
Hooks in Cucumber are methods that run before or after scenarios or features. I use @Before and @After hooks to set up and tear down test environments, such as initializing WebDriver or cleaning up test data.

**3. How do you manage reusability in step definitions?**  
I ensure reusability by creating generic step definitions that can be reused across multiple scenarios. For example, a step definition like “When I send a POST request” can be reused across different API tests. Additionally, I organize my step definitions into classes based on functionality to maintain a clean structure.

**CI/CD & Tools**

**1. How did you integrate your test automation with Jenkins?**  
I integrated my test automation with Jenkins by configuring Jenkins jobs to trigger the execution of automated test scripts whenever there is a change in the code repository. I set up Jenkins to pull the latest code from Git, build the project, and run the tests. The test results are then reported back to Jenkins for visibility.

**2. What’s your Git branching strategy in test automation projects?**  
I follow a **feature-branching** strategy, where each new feature or bug fix gets its own branch. This ensures that changes to the test scripts don’t affect the main branch until they are ready to be merged. I also use **GitFlow** for more structured branching and releases.

**3. Have you worked with Jenkins pipelines? Declarative or scripted?**  
Yes, I’ve worked with Jenkins pipelines, both declarative and scripted. I prefer declarative pipelines for their readability and simplicity, as they allow me to define stages for build, test, and deployment in a structured manner.

**JMeter & Performance Testing**

**1. How did you configure JMeter for load testing?**  
I configure JMeter by creating test plans with multiple threads to simulate concurrent users. I use HTTP request samplers to simulate different API calls and add listeners to analyze the results, such as throughput, response times, and error rates. I also configure assertions to validate that the server responds within acceptable limits.

**2. What performance metrics do you usually analyze?**  
I typically analyze response time, throughput, error rate, and latency. These metrics help me identify performance bottlenecks and areas where the system can be optimized.

**3. How do you identify bottlenecks using JMeter results?**  
I analyze the JMeter results by reviewing the response times and throughput for each request. A significant increase in response time or a decrease in throughput could indicate a performance bottleneck. I also look for high error rates, which might point to issues in the backend systems.

**General QA Process**

**1. How do you prioritize test cases in Agile sprints?**  
In Agile sprints, I prioritize test cases based on feature criticality, risk assessment, and the likelihood of defects. I focus on automating regression tests first, followed by new features. I also prioritize high-risk features that are crucial to the business.

**2. What’s your defect tracking process in JIRA?**  
I log defects in JIRA, providing detailed steps to reproduce, environment details, and severity. I work closely with developers to ensure timely resolution, and I update the status of the defect as it moves through the testing lifecycle.

**3. How do you ensure test coverage for a new feature?**  
I ensure test coverage for a new feature by first understanding the requirements and then creating test cases that cover all scenarios, including boundary cases, edge cases, and negative tests. I also collaborate with developers to ensure that the automated tests provide complete coverage.

**Communication/Soft Skills Assessment Prompts**

**1. Walk me through one of your recent challenging bugs and how you solved it.**  
I recently encountered a bug where the UI elements were not being detected due to timing issues. After investigating, I realized the elements were loading asynchronously. I implemented explicit waits to handle dynamic loading, ensuring that the elements were fully visible before interacting with them.

**2. Explain your current project and your role in it to a non-technical person.**  
In my current project, I am responsible for automating the testing of a web application that helps businesses manage their customer data. My role involves writing scripts that automatically test whether the features are working as expected, reducing the time and effort spent on manual testing.

**3. If you had to onboard a new QA team member, how would you explain your framework?**  
I would start by explaining the structure of the framework, including the use of Page Object Model (POM) for UI testing and the integration of Rest Assured for API testing. I would demonstrate how the tests are organized into feature files (for BDD) and step definitions, and how to use TestNG for test execution. Finally, I would explain how the tests are integrated into the CI/CD pipeline with Jenkins for automation.

**4. How do you stay up to date with the latest tools in test automation?**  
I stay up to date with the latest tools by reading blogs, participating in online communities like Stack Overflow, attending webinars, and experimenting with new tools. I also make sure to keep track of industry trends and explore new releases for tools like Selenium, Cucumber, and Jenkins.