CTS Selenium Interview Questions with Answers Part 5 (2026) | Saravana Tech

84 Elaborate Answers with Code Examples

This guide follows the supplied 84 numbered prompts. Code is illustrative and must be adapted to the actual application DOM, framework, database, and project configuration. Where a prompt lacks a specific scenario or site markup, the answer states that limitation rather than inventing exact details.

How to use this guide

- Personalize project, client, team, tool, and experience answers with facts from your own work.
- Use approved locators, URLs, database names, and credentials for the environment.
- Use explicit waits and stable locators; avoid catching and ignoring errors without a recovery plan.
- Never commit passwords or production credentials to Git.

Contents

1. Frames and inline frames (iframes) handling . . . 5
2. SQL and JDBC connection establishment . . . 5
3. How to handle a dynamic web table . . . 5
4. What does @Test do in TestNG? . . . 6
5. Take a screenshot in Selenium . . . 6
6. How to underline an element . . . 6
7. Highlight text and get foreground/background colour . . . 6
8. Handle Windows pop-up using Robot class . . . 7
9. How to achieve cross-browser testing . . . 7
10. Get tomorrow's date in dd-MM-yyyy format . . . 7
11. Click an element using JavaScriptExecutor . . . 8
12. Difference between Ant and Maven . . . 8
13. When to use Excel versus CSV as test input? Why? . . . 8
14. Web services basics . . . 8
15. TestNG invocationCount and threadPoolSize . . . 8
16. Enumeration, Iterator, and ListIterator . . . 9
17. Generate random strings and numbers . . . 9
18. Get the number of checkboxes . . . 9
19. Types of CSS selectors . . . 9
20. String, StringBuilder, and StringBuffer . . . 9
21. Multiple inheritance using interfaces . . . 9
22. 1000 test cases with defect leakage: strategy . . . 10
23. Scenarios for testing Microsoft Excel using Selenium . . . 10
24. Compare web page data with Excel data . . . 10
25. How do you estimate effort? . . . 10
26. Which version-control tool do you use and what is the process? . . . 10
27. Did the client agree to ExtentReports? . . . 11
28. Scrum meetings and what do you discuss? . . . 11
29. Innovations in Selenium . . . 11
30. Excel is slow—why use it for test data? . . . 11
31. Have you worked on test plan and test strategy? . . . 11
32. How do you generate failed test cases? . . . 12
33. Capture screenshots only for failed test cases . . . 12
34. Button enabled initially and disabled after some time—how verify? . . . 12
35. Retrieve values from a HashMap . . . 12
36. Right-click the mouse . . . 13

37. How do you verify a highlighted element? . . . 13
38. JavaScriptExecutor drawbacks and alternatives . . . 13
39. How do you check a hidden element? . . . 13
40. Do you know database connections? . . . 13
41. Execute against four environments using Maven . . . 14
42. Assert versus verify . . . 14
43. Explain BDD framework . . . 14
44. How to establish database connection . . . 14
45. Explain XPath and its types . . . 14
46. Scenario-based questions . . . 15
47. Read and write data from Excel . . . 15
48. Why is Cucumber preferred? . . . 15
49. Exceptions faced in Cucumber . . . 15
50. What keywords are used in Cucumber? . . . 15
51. Handle alerts and take a screenshot of an alert . . . 15
52. Reporting tool and code . . . 16
53. Jenkins use and scheduling . . . 16
54. Java code to check a prime number . . . 16
55. GitHub questions and SQL query for highest salary . . . 16
56. TestNG annotations and attributes . . . 17
57. Project domain, application, and framework . . . 17
58. Why create a Maven project? . . . 17
59. How Jenkins is used in your office . . . 17
60. Cucumber framework and mandatory keywords . . . 17
61. Convince a customer to select an automation tool . . . 17
62. Difference between git pull and git fetch . . . 18
63. What is wait and which wait is preferred? . . . 18
64. What is POM and why prefer it? . . . 18
65. isDisplayed versus isEnabled . . . 18
66. Drag and drop code . . . 18
67. Rate yourself in SQL . . . 18
68. SQL sorting . . . 18
69. Cross-browser testing . . . 19
70. DB2 database connectivity code . . . 19
71. Day-to-day responsibilities . . . 19
72. Selenium challenges and selecting automatable cases . . . 19
73. Why move from manual to automation? . . . 19

74. Relative and absolute XPath for an IRCTC login page 19
75. Jenkins and GitHub 20
76. Roles and responsibilities in your project 20
77. Which test cases will you automate? 20
78. How many test cases can you execute in a day? 20
79. Resolve a Git merge conflict 20
80. Languages usable with Jenkins besides Java 20
81. Why not run code directly in Jenkins without Maven and GitHub? 20
82. Scenario questions and team management 21
83. Explain an automation plan to a client 21
84. You are in Sprint 2 and receive a client issue from Sprint 1 21

1. Frames and inline frames (iframes) handling

A frame/iframe is a browsing context inside a page. Selenium can switch into it by index, name/id, or a WebElement, then return with `defaultContent()` or `parentFrame()`. Prefer stable id/name or locate the iframe element; use explicit waits.

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(
        By.cssSelector("iframe#paymentFrame")));
driver.findElement(By.id("cardNumber")).sendKeys("4111111111111111");
driver.switchTo().defaultContent();
```

If the iframe has no id or name, locate it as a WebElement first:

```
WebElement frame = driver.findElement(By.cssSelector("iframe.payment-frame"));
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(frame));
```

Note: Nested frames: switch into the parent first, then the child. An inline frame uses the same Selenium APIs.

2. SQL and JDBC connection establishment

JDBC is Java's API for connecting to relational databases. Add the correct JDBC driver dependency, connect using a JDBC URL, use PreparedStatement for parameters, read ResultSet, and close resources with try-with-resources. Keep credentials outside source code.

```
String url = "jdbc:postgresql://localhost:5432/companydb";
String sql = "SELECT employee_id, name FROM employee WHERE department_id = ?";

try (Connection con = DriverManager.getConnection(url, user, password);
     PreparedStatement ps = con.prepareStatement(sql)) {

    ps.setInt(1, 10);
    try (ResultSet rs = ps.executeQuery()) {
        while (rs.next()) {
            System.out.println(rs.getInt("employee_id") + " " + rs.getString("name"));
        }
    }
} catch (SQLException e) {
    throw new RuntimeException("Database query failed", e);
}
```

Note: Use the database-specific URL and driver for Oracle, MySQL, DB2, etc.

3. How to handle a dynamic web table

Locate rows and cells, identify a row by stable business data, and interact within that row rather than relying on row numbers. For pagination, handle each page; for changing DOMs, re-find elements after updates.

```
List<WebElement> rows = driver.findElements(By.cssSelector("table tbody tr"));

for (WebElement row : rows) {
    List<WebElement> cells = row.findElements(By.tagName("td"));
    if (cells.size() >= 3 && cells.get(0).getText().trim().equals("E102")) {
        String employeeName = cells.get(1).getText().trim();
        row.findElement(By.xpath(".//button[normalize-space()='Edit']")).click();
        System.out.println(employeeName);
        break;
    }
}
```

Note: Adjust column indexes and selectors to the actual table.

4. What does @Test do in TestNG?

`@Test` marks a method or class as a TestNG test. TestNG discovers it, executes it according to suite configuration and attributes, records pass/fail/skip, and integrates with dependencies, groups, data providers, and parallel execution.

```
@Test(priority = 1, groups = {"smoke"}, timeOut = 5000)
public void loginTest() {
    Assert.assertEquals(driver.getTitle(), "Home");
}
```

Common attributes include priority, groups, `dependsOnMethods`, `dependsOnGroups`, `invocationCount`, `threadPoolSize`, `dataProvider`, enabled, `timeOut`, `expectedExceptions`, and `alwaysRun` (availability and semantics depend on the TestNG version).

5. Take a screenshot in Selenium

Use TakesScreenshot and save the bytes to a file. Use a unique filename and create the destination folder if needed.

```
File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
Path dest = Paths.get("target", "screenshots", "login-failure.png");
Files.createDirectories(dest.getParent());
Files.copy(src.toPath(), dest, StandardCopyOption.REPLACE_EXISTING);
```

Note: For failure-only screenshots, capture in a TestNG listener's `onTestFailure`(ITestResult result), using that test's WebDriver instance.

6. How to underline an element

Selenium does not have a built-in underline method. If the goal is visual highlighting during debugging, use JavaScript to set `textDecoration`. This changes the page DOM style and is usually for debugging, not functional assertions.

```
WebElement element = driver.findElement(By.id("username"));
((JavascriptExecutor) driver).executeScript(
        "arguments[0].style.textDecoration='underline';", element);
```

Note: For actual UI behavior, verify the CSS property with `getCssValue`("text-decoration-line") or inspect the application's expected styling.

7. Highlight text and get foreground/background colour

You can temporarily set CSS styles through JavaScript for debugging, and read CSS values with `getCssValue`. Note that changing the style is not proof the application naturally highlights the element; verify expected style before modifying it.

```
WebElement el = driver.findElement(By.id("status"));

String textColor  = el.getCssValue("color");
String background = el.getCssValue("background-color");

((JavascriptExecutor) driver).executeScript(
        "arguments[0].style.backgroundColor='yellow'; arguments[0].style.color='black';", el);

System.out.println("Text colour=" + textColor + ", background=" + background);
```

Note: CSS colours often return rgb(...) or rgba(...), not hex.

8. Handle Windows pop-up using Robot class

Robot can send OS-level keyboard/mouse events, but it is fragile and depends on focus, OS, timing, and desktop availability. Prefer WebDriver `sendKeys()` for file inputs and Alert APIs for JavaScript alerts. Robot is mainly a fallback for native dialogs that WebDriver cannot control.

Example for a native file chooser (Windows):

```
Robot robot = new Robot();
robot.delay(1000);

StringSelection path = new StringSelection("C:\\files\\document.pdf");
Toolkit.getDefaultToolkit().getSystemClipboard().setContents(path, null);

robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_V);
robot.keyRelease(KeyEvent.VK_V);
robot.keyRelease(KeyEvent.VK_CONTROL);

robot.keyPress(KeyEvent.VK_ENTER);
robot.keyRelease(KeyEvent.VK_ENTER);
```

Note: Imports include java.awt.* and java.awt.datatransfer.StringSelection. Prefer input[type=file].`sendKeys`(`absolutePath`) when possible.

9. How to achieve cross-browser testing

Parameterize browser selection and create the appropriate WebDriver/Options for Chrome, Firefox, and Edge. Run the same tests against each browser using TestNG parameters, Maven profiles, or a CI matrix. For remote parallel runs, use Selenium Grid and RemoteWebDriver.

Example browser factory:

```
if (browser.equalsIgnoreCase("chrome")) {
    driver = new ChromeDriver(new ChromeOptions());
} else if (browser.equalsIgnoreCase("firefox")) {
    driver = new FirefoxDriver(new FirefoxOptions());
} else if (browser.equalsIgnoreCase("edge")) {
    driver = new EdgeDriver(new EdgeOptions());
} else {
    throw new IllegalArgumentException("Unsupported browser: " + browser);
}
```

Note: Keep tests browser-independent, use stable waits, and record browser/version in reports.

10. Get tomorrow's date in dd-MM-yyyy format

Use LocalDate and DateTimeFormatter; this avoids manual month/year rollover bugs.

```
LocalDate tomorrow = LocalDate.now().plusDays(1);
String formatted = tomorrow.format(DateTimeFormatter.ofPattern("dd-MM-yyyy"));
System.out.println(formatted);
```

For a specific timezone:

```
ZonedDateTime.now(ZoneId.of("Asia/Kolkata")).toLocalDate().plusDays(1);
```

11. Click an element using JavaScriptExecutor

Use JavaScript click only when normal WebDriver click is unsuitable and the reason is understood. A JS click bypasses some user-interaction checks, so it can hide overlay or usability issues.

```
WebElement button = driver.findElement(By.id("submit"));
((JavascriptExecutor) driver).executeScript("arguments[0].click();", button);
```

Note: Prefer wait.until(ExpectedConditions.`elementToBeClickable`(locator)).`click()` first.

12. Difference between Ant and Maven

Ant is a flexible build tool driven by explicitly written XML targets and tasks; it does not impose a standard project lifecycle or built-in dependency management. Maven uses a convention-based lifecycle, POM, dependency management, and plugin ecosystem. Ant offers flexibility; Maven offers standardization and easier dependency handling.

13. When to use Excel versus CSV as test input? Why?

CSV is lightweight, fast, simple, diff-friendly, and suitable for flat tabular data without formatting or multiple sheets. Excel is useful when business users maintain multiple sheets, formulas, cell formatting, or workbook-based templates. Excel parsing has more overhead, so avoid repeatedly opening the workbook for every test; load once or cache parsed data when safe.

Note: Choose based on who maintains the data, structure, size, version-control needs, and required features—not because one format is universally better.

14. Web services basics

A web service exposes functionality over a network. REST commonly uses HTTP methods (GET, POST, PUT, PATCH, DELETE), resources, status codes, headers, and JSON. SOAP is a protocol using XML envelopes and often WSDL contracts. API testing validates status, schema, payload, authentication, error handling, and business rules.

Note: Example: GET /api/employees/10 may return 200 and JSON; POST /api/employees may return 201. Validate positive/negative cases, authorization, response time, and data persistence. Tools include Postman, RestAssured, and SoapUI.

15. TestNG invocationCount and threadPoolSize

`invocationCount` specifies how many times a `@Test` method should run. `threadPoolSize` specifies the number of threads used for those invocations when `invocationCount` creates repeated invocations; `threadPoolSize` is relevant with `invocationCount`. TestNG suite-level parallel execution is configured separately in testng.xml.

```
@Test(invocationCount = 6, threadPoolSize = 3)
public void repeatTest() {
    // test must be thread-safe
}
```

This requests six invocations using a pool of three threads. Avoid shared mutable WebDriver or test data; use per-thread drivers and unique data.

16. Enumeration, Iterator, and ListIterator

Enumeration is a legacy read-only traversal interface, mainly used with Vector/Hashtable; it has `hasMoreElements()`/`nextElement()`. Iterator traverses collections with `hasNext()`/`next()` and may support `remove()`. ListIterator works only with List and supports forward/backward traversal, indexes, `add()`, `set()`, and `remove()`.

Note: Use Iterator for general collection traversal; ListIterator when you need bidirectional List traversal or controlled replacement/insertion.

17. Generate random strings and numbers

Use SecureRandom for security-sensitive values and ThreadLocalRandom/Random for ordinary test data. UUID is convenient for unique IDs. Avoid relying on random values without logging them or controlling reproducibility.

```
String unique = "test_" + UUID.randomUUID().toString().replace("-", "").substring(0, 8);
int number = ThreadLocalRandom.current().nextInt(1000, 10000);   // 1000..9999
```

Note: For deterministic tests, use a seeded Random or a known fixture.

18. Get the number of checkboxes

Find checkbox inputs and count them. If only visible checkboxes should count, filter by `isDisplayed()`.

```
List<WebElement> boxes = driver.findElements(By.cssSelector("input[type='checkbox']"));
System.out.println("Total checkboxes=" + boxes.size());

long visible = boxes.stream().filter(WebElement::isDisplayed).count();
System.out.println("Visible checkboxes=" + visible);
```

19. Types of CSS selectors

Common CSS selectors include tag (input), ID (#username), class (.primary), attribute ([type='email']), compound (input.required), descendant (form input), child (ul > li), multiple selector (h1, h2), prefix/suffix/contains attributes ([id^='user'], [id$='name'], [class*='active']), and pseudo-classes such as :first-child and :nth-of-`type()`. Selenium supports CSS selectors supported by the browser; not all CSS features are locator features in every context.

20. String, StringBuilder, and StringBuffer

String is immutable; changes create new String objects. StringBuilder is mutable and generally faster for repeated concatenation in a single thread. StringBuffer is mutable and synchronizes its methods, making it thread-safe at method level but usually slower than StringBuilder. Use String for fixed text, StringBuilder for building text in loops, and StringBuffer only when its synchronization semantics are needed.

21. Multiple inheritance using interfaces

Java classes cannot extend multiple classes, but can implement multiple interfaces. This provides multiple inheritance of type and allows a class to implement separate contracts.

```
interface Printable { void print(); }
interface Scannable { void scan(); }

class MultiFunctionPrinter implements Printable, Scannable {
    public void print() { System.out.println("Printing"); }
    public void scan()  { System.out.println("Scanning"); }
```

```
}
```

Note: If two interfaces provide conflicting default methods, the class must resolve the conflict explicitly.

22. 1000 test cases with defect leakage: strategy

Do not simply rerun all tests without analysis. Triage leaked defects by severity, customer impact, and escape point; perform root-cause analysis; map gaps to requirements, risk areas, and test layers; add missing tests; strengthen code reviews/static/API/integration checks; stabilize environments and test data; prioritize automation for repeatable high-risk regression; and track escape rate and recurrence.

Note: A practical plan: 1) classify escaped defects; 2) identify why current tests missed them; 3) add targeted regression cases; 4) prioritize critical business journeys; 5) integrate smoke/regression into CI; 6) monitor leakage trend and flaky-test rate. 1000 tests alone do not guarantee coverage.

23. Scenarios for testing Microsoft Excel using Selenium

Selenium automates the web application interface, not the native Excel desktop app itself. If Excel is accessed through a web application such as Excel for the web, test UI workflows like opening a workbook, editing cells, saving, sharing, filtering, and validating messages. For workbook file content, use Apache POI or another file-processing library instead of Selenium.

Note: Possible web UI cases: open workbook, edit a cell, apply filter, save changes, verify permission behavior, validate error banners, and confirm concurrent-edit messages. Avoid using Selenium to inspect a local .xlsx file's internal values.

24. Compare web page data with Excel data

Read expected values from Excel with Apache POI, read actual values from the page, normalize whitespace/case/date formats where appropriate, and compare row by row. Report all mismatches with row/column details rather than stopping at the first one.

Example outline:

```
Map<String, String> expected = readExcel();   // Apache POI
Map<String, String> actual   = readPage();    // Selenium
Assert.assertEquals(actual, expected, "Page data differs from workbook");
```

Note: For large tables, key records by a unique ID and compare selected columns; ensure both sources are from the same data snapshot.

25. How do you estimate effort?

Break work into activities: requirement analysis, test design, test data/environment, automation development, reviews, execution, defect triage, maintenance, and reporting. Estimate complexity, dependencies, uncertainty, team capacity, and historical velocity; include contingency for risks.

Note: In Agile, estimate stories collaboratively using story points or a team-agreed method. For test effort, use analogous past work or bottom-up task estimates and revise as uncertainty reduces. State assumptions and exclusions.

26. Which version-control tool do you use and what is the process?

If using Git, a typical workflow is pull/fetch, create or switch to a feature branch, modify code, run tests, inspect git status/diff, add and commit, push, create a pull request, get review, resolve conflicts, and merge through the team's policy.

```
git switch -c feature/login-test
git status
git add src/test/java
git commit -m "Add login test"
git push -u origin feature/login-test
```

Note: Follow branch protections and never commit credentials.

27. Did the client agree to ExtentReports?

Answer based on actual project governance. Explain how reporting requirements were discussed—readability, pass/fail/skip, screenshots, logs, trends, access, and CI publishing—and who approved the tool. If the project uses a different tool, say so honestly.

Note: Do not claim client approval unless it happened. A safe template: “We evaluated reporting against the project’s requirements and followed the client/team-approved tool. The report includes …”.

28. Scrum meetings and what do you discuss?

Daily Scrum focuses on progress toward the Sprint Goal and adapting the plan. Sprint Planning selects work and sets a goal; refinement clarifies upcoming stories; Review demonstrates the increment and gathers feedback; Retrospective identifies process improvements. QA may raise blockers, test progress, risks, defects, and readiness, keeping detailed troubleshooting outside the daily meeting.

29. Innovations in Selenium

Examples of useful improvements include reusable locator utilities, smart explicit-wait wrappers, automatic failure screenshots, parallel-safe driver management, data factories, flaky-test analytics, CI smoke gates, API setup to reduce UI setup time, and accessibility checks. Selenium 4 also supports relative locators, improved Grid architecture, and Chrome DevTools Protocol integrations in supported setups.

Note: Describe an innovation you personally implemented, the problem it solved, and the evidence of improvement. Avoid calling ordinary framework setup an innovation without context.

30. Excel is slow—why use it for test data?

Excel is convenient for business-owned data but may be slower than in-memory fixtures, JSON, or CSV, especially if reopened for each test. Read the workbook once per suite/data provider, cache immutable test data when appropriate, avoid shared mutable workbook objects across parallel threads, and use CSV/JSON/database fixtures for large or machine-maintained datasets.

Note: The format should fit the data owner and workflow. Performance issues are usually about repeated I/O and design, not simply the .xlsx extension.

31. Have you worked on test plan and test strategy?

Explain only the work you performed. A test strategy is the higher-level approach: scope, levels, tools, risks, automation, environments, and quality objectives. A test plan applies that approach to a project/release: scope, schedule, roles, estimates, entry/exit criteria, deliverables, and risks. In some organizations the documents are combined.

32. How do you generate failed test cases?

In TestNG, failed tests appear in TestNG/Surefire results. You can use an ITestListener to capture failure details, or rerun failures through a generated failed-suite XML when supported by the TestNG execution flow. For Maven, inspect target/surefire-reports and configure CI to publish test results.

Example listener:

```
public void onTestFailure(ITestResult result) {
    System.out.println("Failed: " + result.getName() + " : " + result.getThrowable());
}
```

Note: Do not confuse reporting failed tests with automatically rerunning them; retry analyzers/listeners require careful configuration.

33. Capture screenshots only for failed test cases

Implement TestNG ITestListener.`onTestFailure` and obtain the WebDriver instance associated with the failed test/thread. Save a screenshot with a unique test name and timestamp. A ThreadLocal<WebDriver> driver factory is helpful for parallel execution.

```
public void onTestFailure(ITestResult r) {
    WebDriver d = DriverManager.getDriver();
    if (d != null) {
        File f = ((TakesScreenshot) d).getScreenshotAs(OutputType.FILE);
        Path dest = Paths.get("target/screenshots",
                r.getName() + "-" + System.currentTimeMillis() + ".png");
        try {
            Files.createDirectories(dest.getParent());
            Files.copy(f.toPath(), dest);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Note: Add the required imports and ensure DriverManager returns the correct thread's driver.

34. Button enabled initially and disabled after some time—how verify?

Verify the initial state, trigger or wait for the transition condition, then assert the disabled state. Use explicit waits instead of Thread.sleep when possible.

```
WebElement btn = driver.findElement(By.id("submit"));
Assert.assertTrue(btn.isEnabled());

new WebDriverWait(driver, Duration.ofSeconds(10))
        .until(d -> !d.findElement(By.id("submit")).isEnabled());

Assert.assertFalse(btn.isEnabled());
```

Note: If the button disables only after an action or timer, reproduce that prerequisite first.

35. Retrieve values from a HashMap

Use get(key), `keySet()`, `values()`, `entrySet()`, or `getOrDefault()`. Iterating `entrySet` is efficient when both keys and values are needed.

```
Map<String, Integer> scores = new HashMap<>();
scores.put("Asha", 90);

for (Map.Entry<String, Integer> e : scores.entrySet()) {
```

```
    System.out.println(e.getKey() + "=" + e.getValue());
}
```

Note: HashMap does not guarantee iteration order.

36. Right-click the mouse

Use Actions.`contextClick`(element).`perform()`.

```
WebElement row = driver.findElement(By.id("row1"));
new Actions(driver).contextClick(row).perform();
```

Note: For a custom HTML context menu, then locate and click the menu item. Browser-native menu items are not generally accessible via WebDriver.

37. How do you verify a highlighted element?

Read the relevant CSS properties, such as border-color, background-color, outline-color, or box-shadow, and compare with the expected computed style. A screenshot can help visually, but assertions should target the intended property.

```
String bg = element.getCssValue("background-color");
Assert.assertEquals(bg, "rgba(255, 255, 0, 1)");
```

Note: Actual formats and colours depend on the browser and computed CSS; inspect the real expected value.

38. JavaScriptExecutor drawbacks and alternatives

JavaScriptExecutor can bypass normal user interaction checks, hide real UI problems, and make tests less representative of user behavior. Prefer WebDriver click/`sendKeys`/actions and explicit waits first. JS is useful for reading DOM properties, scrolling, or special cases where normal interaction is impossible and the reason is documented.

39. How do you check a hidden element?

Use `findElements` to determine DOM presence, `isDisplayed()` to check visibility, and `getAttribute`/style only if needed. A hidden element can exist in the DOM while `isDisplayed()` returns false. If it is not in the DOM, `findElements` returns an empty list.

```
List<WebElement> els = driver.findElements(By.id("message"));

if (els.isEmpty()) {
    System.out.println("Not in DOM");
} else {
    System.out.println("Present; displayed=" + els.get(0).isDisplayed());
}
```

Note: Do not use JS to interact with hidden controls unless the test explicitly validates hidden-state behavior.

40. Do you know database connections?

Yes, if true: explain JDBC, driver dependency, connection URL, credentials, PreparedStatement, ResultSet, transactions, and closing resources. Use database queries to validate persisted state where permitted, and avoid using direct DB checks as a substitute for all UI behavior.

41. Execute against four environments using Maven

Use Maven profiles or system properties to select environment configuration, and pass the profile/property in each run. Avoid duplicating test code. Keep environment URLs and secrets in configuration/CI variables, not source code.

Pass the environment as a system property:

```
mvn test -Denv=qa1
mvn test -Denv=qa2
```

Note: A profile can set environment-specific properties; alternatively execute four CI matrix jobs. Do not run four concurrent suites against shared data unless tests are isolated.

42. Assert versus verify

A hard assertion stops the test method at the first failed assertion. A soft assertion records failures and allows later checks to run; in TestNG, call `SoftAssert.assertAll()` at the end or the failures will not fail the test.

```
SoftAssert soft = new SoftAssert();
soft.assertEquals(actualTitle, expectedTitle);
soft.assertTrue(saveButton.isDisplayed());
soft.assertAll();   // required - reports all collected failures
```

43. Explain BDD framework

Behavior-Driven Development expresses behavior in business-readable Given/When/Then scenarios. With Cucumber, Feature files contain Gherkin, step definitions map steps to code, hooks manage setup/teardown, and page objects/utilities implement actions. BDD works best when scenarios are collaborative and express business behavior rather than low-level click sequences.

Example scenario:

```
Scenario: Successful login
  Given I am on the login page
  When I sign in with valid credentials
  Then I should see the dashboard
```

44. How to establish database connection

Use JDBC with the correct driver and URL; create a Connection, use PreparedStatement, process ResultSet, and close resources. Use secure credentials and least-privilege access. See Question 2 for a full example.

45. Explain XPath and its types

Common interview categories are absolute XPath (/html/body/...) and relative XPath (//input[`@id`='user']). XPath expressions can use attributes, text, predicates, functions, and axes. Examples: //button[`@type`='submit']; //a[normalize-`space()`='Home']; //input[contains(`@id`,'user')]; //label[.='Email']/following::input[1]. Prefer stable relative locators.

```
//button[@type='submit']
//a[normalize-space()='Home']
//input[contains(@id,'user')]
//label[.='Email']/following::input[1]
```

46. Scenario-based questions

The prompt does not include a specific scenario. For any scenario, clarify expected behavior, initial state, data, locator/DOM constraints, and error handling; then state the approach and code. Do not claim an exact answer where the actual scenario is missing.

47. Read and write data from Excel

Use Apache POI. Read with WorkbookFactory/XSSFWorkbook and DataFormatter; write by creating/updating cells and calling workbook.write(OutputStream). Always close workbook and streams, and avoid concurrent writes to the same workbook.

Example write:

```
try (Workbook wb = new XSSFWorkbook();
     OutputStream out = Files.newOutputStream(Path.of("target/result.xlsx"))) {

    Sheet s = wb.createSheet("Results");
    s.createRow(0).createCell(0).setCellValue("Status");
    s.getRow(0).createCell(1).setCellValue("PASS");
    wb.write(out);
}
```

Note: For existing files, open the workbook from an input stream and write to an output stream carefully.

48. Why is Cucumber preferred?

Cucumber can make business behavior readable to non-technical stakeholders and supports collaboration through Gherkin scenarios, reusable steps, tags, hooks, and reports. It is not automatically better for every project: it adds step-definition maintenance and can become verbose if scenarios describe UI mechanics instead of behavior.

49. Exceptions faced in Cucumber

Common issues include UndefinedStepException/undefined steps, ambiguous step definitions, duplicate glue matches, failed hooks, parameter conversion errors, and assertion/runtime exceptions inside step definitions. Exact exception names vary by Cucumber version. Fix by aligning feature wording with step definitions, avoiding overlapping expressions, configuring glue paths, and preserving failure stack traces.

50. What keywords are used in Cucumber?

Gherkin keywords include Feature, Rule, Scenario, Scenario Outline, Examples, Given, When, Then, And, But, Background, and doc strings/data tables. Some are structural, and step keywords express context/action/outcome. The exact supported syntax depends on Gherkin version.

51. Handle alerts and take a screenshot of an alert

For JavaScript alerts, switch to Alert, read text, and accept/dismiss. Selenium screenshots generally capture the browser viewport but native alert rendering can be browser-dependent; screenshot before interacting may or may not include the alert. Capture a screenshot of the page and record `alert.getText()` for reliable evidence.

```
Alert a = new WebDriverWait(driver, Duration.ofSeconds(10))
        .until(ExpectedConditions.alertIsPresent());

String message = a.getText();
```

```
// optionally take a screenshot before accepting
File f = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
a.accept();
```

Note: Do not expect Selenium to screenshot an OS-level dialog consistently.

52. Reporting tool and code

Name the tool actually used (for example ExtentReports, Allure, or built-in TestNG/Surefire reports). A TestNG listener can attach screenshots and failure messages; reporting APIs vary by version.

Illustrative ExtentReports pattern:

```
ExtentReports extent = new ExtentReports();
ExtentTest test = extent.createTest("Login test");
test.pass("Login successful");
extent.flush();
```

Note: Configure a report destination and ensure flush/cleanup occurs even when tests fail. Do not claim client approval unless confirmed.

53. Jenkins use and scheduling

Jenkins automates checkout, build, test execution, report publishing, artifact archiving, and notifications. Schedule a job through Build Triggers/cron or use a Pipeline Jenkinsfile with a cron trigger. Jenkins cron syntax uses five fields; H/ notation spreads load across time.

Example Jenkinsfile snippet:

```
pipeline {
    agent any
    triggers { cron('H 2 * * 1-5') }
    stages {
        stage('Test') {
            steps { sh 'mvn clean test' }
        }
    }
}
```

Note: Use bat instead of sh on Windows agents. Configure credentials securely and publish test reports.

54. Java code to check a prime number

Check numbers below 2 as non-prime; test divisors only up to the square root.

```
static boolean isPrime(int n) {
    if (n < 2) return false;
    for (int i = 2; i <= n / i; i++) {
        if (n % i == 0) return false;
    }
    return true;
}

System.out.println(isPrime(17));   // true
```

55. GitHub questions and SQL query for highest salary

GitHub hosts Git repositories and adds collaboration features such as pull requests, reviews, issues, and Actions. Git is the version-control tool; GitHub is a hosting/collaboration platform. To fetch the employee(s) with the highest salary, use a subquery to preserve ties.

```
SELECT employee_id, employee_name, salary
```

```
FROM   employees
WHERE  salary = (SELECT MAX(salary) FROM employees);
```

Note: To fetch one row only, use ORDER BY salary DESC with database-specific limiting syntax, but that may discard ties.

56. TestNG annotations and attributes

Core lifecycle annotations: `@BeforeSuite`/`@AfterSuite`, `@BeforeTest`/`@AfterTest`, `@BeforeClass`/`@AfterClass`, `@BeforeMethod`/`@AfterMethod`, `@BeforeGroups`/`@AfterGroups`. `@Test` attributes include priority, groups, `dependsOnMethods`/Groups, `invocationCount`, `threadPoolSize`, `dataProvider`, enabled, `timeOut`, `expectedExceptions`, and `alwaysRun`. `@DataProvider` supplies data; `@Parameters` reads XML parameters; `@Factory` creates test instances; `@Listeners` registers listeners. Exact behavior depends on TestNG version and configuration.

57. Project domain, application, and framework

Explain the real business domain, main workflows, users, architecture at a high level, your role, tools, framework layers, CI/reporting, and key quality risks. Do not disclose confidential client or production details. Use a real example to show how a requirement becomes a test and report.

58. Why create a Maven project?

Maven standardizes build structure and lifecycle, resolves dependencies, manages plugins/versions, and makes local/CI execution repeatable. A pom.xml documents project coordinates, dependencies, properties, and plugins. Maven is not mandatory for Java, but it reduces manual build and JAR management.

59. How Jenkins is used in your office

Describe actual pipeline steps: source checkout, build, test execution, report publication, screenshots/log archiving, and notifications. Mention schedule or trigger type only if you know it. If you have only run jobs rather than maintained pipelines, state that distinction.

60. Cucumber framework and mandatory keywords

A feature typically starts with Feature and contains Scenario/Scenario Outline; Given/When/Then steps describe context, action, and outcome. Examples is used with Scenario Outline. Some keywords such as Rule, Background, And, and But are optional depending on scenario structure. Step definitions and glue code are required for executable matching steps.

61. Convince a customer to select an automation tool

Start with requirements and constraints: application technology, browser/platform coverage, team skills, licensing, integrations, security, maintainability, reporting, CI, and total cost. Run a small proof of concept on representative high-risk workflows and compare stability, execution time, maintainability, and support. Recommend the tool based on evidence and customer priorities, not preference alone.

62. Difference between git pull and git fetch

git fetch downloads remote refs/commits without merging them into the current branch. git pull performs fetch followed by integration (merge or rebase, depending on configuration/flags). Fetch lets you inspect changes first; pull updates the current branch. Teams should follow their configured strategy.

63. What is wait and which wait is preferred?

Waits synchronize tests with asynchronous UI behavior. Explicit waits are generally preferred for dynamic UI because they wait for a specific condition, such as visibility or clickability. Implicit wait applies globally to element lookup; FluentWait customizes polling/ignored exceptions. Avoid fixed sleeps except for rare diagnostic cases.

64. What is POM and why prefer it?

Page Object Model puts page locators and actions in page/component classes, leaving tests to express business scenarios and assertions. It reduces duplication and centralizes maintenance when UI locators change. It is a design pattern, not a test runner or Maven POM file.

65. isDisplayed versus isEnabled

`isDisplayed()` indicates whether the element is rendered/visible according to WebDriver. `isEnabled()` indicates whether the control is enabled for interaction, typically relevant to form controls. An element can be visible but disabled; visibility and enabled state are different.

66. Drag and drop code

Use Actions.`dragAndDrop`(source,target).`perform()`.

```
WebElement source = driver.findElement(By.id("drag"));
WebElement target = driver.findElement(By.id("drop"));
new Actions(driver).dragAndDrop(source, target).perform();
```

Note: Some HTML5/custom widgets need `clickAndHold`/`moveToElement`/release or application-specific events; verify the resulting state.

67. Rate yourself in SQL

Give an honest rating supported by actual skills: SELECT, WHERE, joins, GROUP BY, aggregate functions, subqueries, insert/update, and test-data validation. Mention what you can do independently and what you are learning.

68. SQL sorting

Use ORDER BY. ASC is ascending and DESC descending. Without ORDER BY, result order is not guaranteed.

```
SELECT employee_name, salary
FROM   employees
ORDER BY salary DESC, employee_name ASC;
```

69. Cross-browser testing

Run the same functional scenarios across required browser/OS combinations. Parameterize browser creation, use compatible browser options, isolate data, and record browser versions. Grid/CI matrices can parallelize runs; focus coverage based on customer usage and known browser differences.

70. DB2 database connectivity code

Use the IBM DB2 JDBC driver and a DB2 JDBC URL. Exact URL, driver dependency, authentication, and TLS settings depend on your DB2 deployment.

```
String url = "jdbc:db2://db-host:50000/DBNAME";
String sql = "SELECT EMPNO FROM EMPLOYEE FETCH FIRST 10 ROWS ONLY";

try (Connection con = DriverManager.getConnection(url, user, password);
     PreparedStatement ps = con.prepareStatement(sql);
     ResultSet rs = ps.executeQuery()) {

    while (rs.next()) {
        System.out.println(rs.getString(1));
    }
}
```

Note: Add the IBM DB2 JDBC driver dependency and use approved credentials. Verify syntax against your DB2 version.

71. Day-to-day responsibilities

Describe your actual day: review requirements and stories, attend refinement/planning/daily meetings, design tests, maintain automation, execute suites, analyze failures, raise/triage defects, review code, update reports, and coordinate release readiness. Tailor it to your role and avoid claiming tasks you do not perform.

72. Selenium challenges and selecting automatable cases

Common challenges include dynamic DOMs, timing/flaky tests, unstable test data, browser differences, CAPTCHA/OTP constraints, environment instability, and slow suites. Automate stable, repeatable, high-value regression and data-heavy scenarios; avoid automating one-off, rapidly changing, subjective visual, or blocked third-party flows without a clear ROI. Keep exploratory testing manual and complement automation.

73. Why move from manual to automation?

Explain that manual testing built your understanding of business workflows and risk, then you identified repeatable regression scenarios where automation improved consistency and feedback speed. Mention your learning path and a real example. Avoid implying manual testing is unnecessary; both approaches serve different purposes.

74. Relative and absolute XPath for an IRCTC login page

Use the actual DOM, not a guessed site structure. Absolute XPath starts at the document root, e.g. /html/body/... and is fragile. Relative XPath starts with // and uses stable attributes/text, e.g. //input[`@id`='`userId`'] or //button[normalize-`space()`='Login']. Since actual current page markup is not provided, an exact IRCTC XPath cannot be guaranteed; inspect the element in browser DevTools.

75. Jenkins and GitHub

GitHub stores version-controlled code and supports pull requests/reviews. Jenkins checks out a branch, builds/tests it, publishes results, and can notify the team. A webhook or polling schedule can trigger Jenkins after pushes. Use credentials securely and branch protections to control merges.

76. Roles and responsibilities in your project

State your real scope: requirement analysis, test planning, automation design/coding, code reviews, framework maintenance, execution, defect triage, CI/reporting, stakeholder updates, and mentoring/coordination. Distinguish work you personally own from team responsibilities.

77. Which test cases will you automate?

Prioritize high-value, repeatable, stable tests: smoke/sanity, critical business paths, regression, data-driven validations, API/UI integration checks, and cross-browser flows with frequent repetition. Avoid automating unstable requirements, one-time scenarios, subjective usability judgments, or expensive setup with little return unless risk justifies it.

78. How many test cases can you execute in a day?

There is no universal number. It depends on test complexity, environment, data setup, execution time, failure analysis, and whether tests are manual or automated. Give a truthful range from your own metrics and distinguish tests executed from tests analyzed and defects triaged.

79. Resolve a Git merge conflict

Run git status to identify conflicted files; open each file and resolve conflict markers; inspect the result, run tests, git add resolved files, then git commit (for merge) or continue the rebase. Do not blindly choose one side without understanding changes.

```
git fetch origin
git merge origin/main
# edit the conflicted files, then:
git add <file>
git commit
```

Note: For a rebase, use git rebase --continue after staging. Coordinate with teammates and avoid force-pushing shared history.

80. Languages usable with Jenkins besides Java

Jenkins can orchestrate builds/tests for many languages, including Java, JavaScript/TypeScript, Python, C#, Go, Ruby, PHP, and shell/PowerShell scripts, depending on installed tools and plugins. Jenkins itself is commonly extended through plugins and pipelines; it is not limited to Java.

81. Why not run code directly in Jenkins without Maven and GitHub?

Jenkins can run scripts directly, and GitHub is not mandatory. Git provides source versioning and collaboration; Maven provides repeatable dependency/build/test lifecycle for Java. Without them, you must manage source retrieval, JARs, classpaths, build steps, and versions manually. For small scripts direct execution can work, but version control/build tools improve reliability and traceability.

82. Scenario questions and team management

Clarify the scenario, identify impact and urgency, gather facts, communicate ownership, agree a plan, track actions, and escalate risks early. For team management, prioritize work transparently, support blockers, review quality, give feedback, and protect focus. Use a real example and measurable result where possible.

83. Explain an automation plan to a client

Start with objectives and success criteria; understand the application, risks, environments, data, browsers, and release cadence; evaluate tools; run a proof of concept; propose framework architecture; prioritize candidates by ROI/risk; define reporting/CI/security; estimate effort and dependencies; pilot, review, and scale. Explain what remains manual and the maintenance cost. Obtain agreement on scope and metrics.

84. You are in Sprint 2 and receive a client issue from Sprint 1

Triage severity, customer impact, reproducibility, and release risk immediately. Create/link a defect or support item, inform the Product Owner/team, and agree whether it is an urgent production fix or can be prioritized in the backlog. For a critical issue, the team may need to replan sprint work with stakeholder agreement; do not silently add work or abandon the Sprint Goal. Add regression coverage after the fix.