Agilisys Selenium Interview Questions with Answers (2026) | Saravana Tech

Elaborate answers with examples for the 47 requested interview topics

Profile personalisation note: sample answers assume around 13 years in software testing and a Java + Selenium + TestNG + Cucumber/POM hybrid framework. Update project details, tools, achievements and metrics to match your actual experience. Do not claim hands-on experience you do not have.

1. Testing basics

1. Testing basics: what should you prepare?

Review fundamentals: verification vs validation, SDLC/STLC, test levels, functional vs non-functional testing, smoke vs sanity, retesting vs regression, severity vs priority, test case vs test scenario, defect life cycle, positive vs negative testing, boundary value analysis, equivalence partitioning and exploratory testing. For each, know a definition and one example. If an interviewer recommends Guru99, use the relevant software-testing tutorial pages as revision material, but confirm that your answers align with the project context.

2. Verification vs validation

Verification asks “Are we building the product right?” and checks work products without necessarily executing the software—for example, requirement reviews and design inspections. Validation asks “Are we building the right product?” and evaluates whether the implemented application meets user needs, commonly through execution-based testing.

3. Smoke vs sanity vs regression vs retesting

Smoke testing is a broad, shallow check that a build is stable enough for deeper testing. Sanity testing is a focused check of a particular change or area. Retesting verifies that a specific reported defect has been fixed. Regression testing checks that existing functionality has not been broken by a change.

4. Severity vs priority

Severity describes the impact of a defect on the system or business. Priority describes how urgently it should be fixed. A crash blocking all users may be high severity and high priority; a typo on a rarely used page may be low severity and low priority. Business context can change priority.

2. SQL and PL/SQL

5. What is GROUP BY?

GROUP BY groups rows that share values so aggregate functions can calculate a result per group. Example: count employees in each department.

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id;
```

6. What is HAVING?

HAVING filters groups after aggregation. WHERE filters individual rows before grouping. Example: show departments with more than five employees.

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 5;
```

7. GROUP BY vs ORDER BY

GROUP BY combines rows into groups for aggregate calculations. ORDER BY sorts the output rows. They can be used together.

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id
ORDER BY employee_count DESC;
```

8. DELETE vs TRUNCATE

DELETE removes selected rows and can use a WHERE clause. TRUNCATE removes all rows from a table and generally cannot filter individual rows. In Oracle, TRUNCATE is DDL and performs an implicit commit; rollback generally cannot restore the truncated rows. DELETE is DML and can be rolled back before commit. Exact locking, logging, identity and trigger behaviour varies by database.

9. What is PL/SQL?

PL/SQL is Oracle's procedural extension to SQL. It supports variables, conditions, loops, exceptions, procedures, functions, packages and triggers. It allows SQL operations and procedural logic to be combined on the database server.

10. PL/SQL example: anonymous block

A basic PL/SQL block has a DECLARE section (optional), BEGIN section (required) and EXCEPTION section (optional).

```
SET SERVEROUTPUT ON;
DECLARE
    v_total NUMBER := 10;
BEGIN
    v_total := v_total + 5;
    DBMS_OUTPUT.PUT_LINE('Total = ' || v_total);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

11. Procedure vs function

A procedure performs an operation and may return values through OUT parameters. A function returns a value using RETURN and can often be used in SQL expressions when it meets Oracle's SQL-call requirements. Both can accept parameters and contain exception handling.

12. PL/SQL cursor

A cursor represents a query result. Implicit cursors are managed by Oracle for statements such as SELECT INTO or DML. Explicit cursors are declared by the developer when processing a query result set row by row. Prefer set-based SQL when it can solve the problem more simply and efficiently.

3. QTP/UFT and Selenium automation

13. What is QTP/UFT?

QTP (QuickTest Professional), now commonly referred to as UFT One in current product naming, is a commercial test automation tool historically associated with functional and regression testing of desktop, web and other supported applications. It has used VBScript-based automation in traditional UFT workflows. Explain only the version and hands-on features you have actually used.

14. QTP/UFT vs Selenium

UFT is a commercial product with vendor-provided support and broad packaged capabilities for supported technologies. Selenium is an open-source browser automation framework with language bindings such as Java and a broad ecosystem. Selenium usually needs the team to assemble its framework, reporting and integrations. The right choice depends on application technology, licensing, team skills, support needs and maintenance cost.

15. What is Selenium?

Selenium is an open-source browser automation suite. WebDriver controls supported browsers; Selenium Grid can distribute tests across machines and browser configurations. It is commonly used for web UI automation and can be integrated with Java, TestNG/JUnit, Cucumber, Maven, reporting tools and CI pipelines.

16. Explain your current framework

Sample answer: “I use a Cucumber + Page Object Model hybrid framework. Feature files describe scenarios in Gherkin, step definitions connect those steps to Java code, and page classes encapsulate locators and page actions. Reusable utilities support waits, dropdowns and configuration. TestNG manages execution where configured, Maven manages dependencies, Git manages source control, and reports/screenshots help investigate failures.” Keep only the components that exist in your real framework.

17. What is POM?

Page Object Model is a design pattern that separates page-specific locators and actions from test logic. It reduces duplicate locators, improves readability and makes UI changes easier to maintain. POM is not a complete framework by itself; configuration, data, assertions, reporting and execution still need to be designed.

18. What is TestNG and why use it?

TestNG is a Java testing framework with annotations, test grouping, dependencies, parameterisation, DataProvider, listeners, parallel execution and XML suite configuration. It is often used with Selenium to organise and execute tests and produce execution results.

19. TestNG annotations, attributes and @BeforeTest

Annotations mark test methods and lifecycle hooks. Common annotations include @Test, @BeforeSuite/@AfterSuite, @BeforeTest/@AfterTest, @BeforeClass/@AfterClass, @BeforeMethod/@AfterMethod, @DataProvider, @Parameters, @Factory and @Listeners. Attributes are settings on annotations, such as priority, groups, dependsOnMethods, enabled, invocationCount and timeOut. @BeforeTest runs before test methods in the relevant testng.xml <test> scope; @BeforeMethod runs before each test method.

20. JUnit vs TestNG

Both are Java test frameworks. TestNG has built-in XML suite configuration, groups, DataProvider, dependencies and flexible parallel execution features that many Selenium teams find useful. JUnit has a strong ecosystem and modern extensions, and newer JUnit versions also support parameterised tests and parallel execution features. Do not say one is universally better; compare the versions and requirements used by the project.

21. How to run three cases in parallel and the next three in series?

Place the first three classes in one TestNG <test> with parallel="classes" and thread-count="3". Place the remaining three in a second <test> with parallel="none". Keep suite-level parallelism disabled so the second group does not run at the same time as the first. Verify the tests do not share unsafe WebDriver or test data.

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="GroupedSuite" parallel="none">
  <test name="FirstThreeParallel" parallel="classes" thread-count="3">
    <classes>
      <class name="tests.Test1"/>
      <class name="tests.Test2"/>
      <class name="tests.Test3"/>
    </classes>
  </test>
```

```
  <test name="NextThreeSequential" parallel="none">
    <classes>
      <class name="tests.Test4"/>
      <class name="tests.Test5"/>
      <class name="tests.Test6"/>
    </classes>
  </test>
</suite>
```

4. Java and OOP concepts

22. Four OOP principles

Encapsulation bundles state and behaviour and controls access to data; abstraction exposes essential behaviour while hiding implementation details; inheritance allows a class to reuse or specialise another class; polymorphism allows the same interface or parent reference to represent different implementations.

23. Explain polymorphism

Compile-time polymorphism is commonly achieved through method overloading, where methods share a name but differ in parameter lists. Runtime polymorphism is achieved through method overriding, where a subclass supplies an implementation selected according to the actual object type.

```
class Animal {
    void sound() { System.out.println("Animal sound"); }
}
class Dog extends Animal {
    @Override
    void sound() { System.out.println("Bark"); }
}
Animal a = new Dog();
a.sound(); // Bark
```

24. Overriding vs overloading

Overloading uses the same method name with different parameter lists, typically in the same class; return type alone cannot distinguish overloads. Overriding replaces an inherited instance method with a compatible implementation in a subclass and supports runtime dispatch. Static methods are hidden rather than overridden, and private methods are not overridden.

25. What is the static keyword?

A static field belongs to the class rather than each object. A static method can be called using the class name and cannot directly access instance members without an object reference. Static initialisation blocks run when the class is initialised, usually once per class loader, before normal use of the class. Static methods are not dynamically overridden.

26. Static block example

A static block is useful for one-time class-level initialisation. Avoid complicated work or swallowed exceptions in static blocks because failures can make class initialisation fail.

```
class Example {
    static int value;
    static {
        value = 100;
        System.out.println("Static block");
    }
    public static void main(String[] args) {
        System.out.println(Example.value);
    }
}
```

27. Why can’t == compare String values reliably?

For objects, == checks whether two references point to the same object, not whether their text is equal. Use equals() for case-sensitive content comparison or equalsIgnoreCase() for case-insensitive comparison. String literals may be interned, so == may appear to work in some examples, but it is not a reliable content comparison.

```
String a = new String("Java");
String b = new String("Java");
System.out.println(a == b);      // false
System.out.println(a.equals(b)); // true
```

28. How do you add two strings?

Use + or concat() for concatenation. For repeated concatenation in a loop, use StringBuilder to avoid creating many intermediate String objects.

```
String first = "Hello";
String second = " Java";
System.out.println(first + second); // Hello Java
System.out.println(first.concat(second));
```

29. How do you declare strings using an array?

A String array stores multiple String references. Arrays have a fixed length after creation.

```
String[] names = {"Arun", "Priya", "Ravi"};
String[] cities = new String[3];
cities[0] = "Chennai";
cities[1] = "Bengaluru";
cities[2] = "Hyderabad";

for (String name : names) {
    System.out.println(name);
}
```

30. Java program: string palindrome

A palindrome reads the same forwards and backwards. This example ignores case and non-alphanumeric characters.

```
public static boolean isPalindrome(String input) {
    if (input == null) return false;
    String s = input.replaceAll("[^A-Za-z0-9]", "")
                    .toLowerCase();
    return s.equals(new StringBuilder(s).reverse().toString());
}
```

31. Java program: Fibonacci series

The Fibonacci sequence begins 0, 1 and each subsequent number is the sum of the previous two. This example prints n terms.

```
int n = 10;
int first = 0, second = 1;
for (int i = 0; i < n; i++) {
    System.out.print(first + " ");
    int next = first + second;
    first = second;
    second = next;
}
```

32. Java program: sum numeric strings using Integer.parseInt

Convert each numeric string to an integer, add it to a running total, and handle invalid values if input may be malformed.

```
String[] values = {"10", "20", "30", "40"};
int sum = 0;
for (String value : values) {
    sum += Integer.parseInt(value);
}
System.out.println(sum); // 100
```

33. Java program: count character occurrences using Map

Use a map from character to count. This example removes spaces and counts case-sensitively; change the normalisation if the requirement differs.

```
String input = "automation";
Map<Character, Integer> counts = new LinkedHashMap<>();
for (char ch : input.toCharArray()) {
    if (ch == ' ') continue;
    counts.put(ch, counts.getOrDefault(ch, 0) + 1);
}
System.out.println(counts);
```

34. Java program: salary tax using if/else

Tax slabs vary by country, tax year and regime. In an interview, clarify whether the task asks for a simple slab classification or actual progressive tax calculation. This is only a simple illustrative threshold example, not tax advice.

```
double salary = 900000;
double rate;
if (salary <= 300000) {
    rate = 0.00;
} else if (salary <= 600000) {
    rate = 0.05;
} else if (salary <= 1000000) {
    rate = 0.10;
} else {
    rate = 0.20;
}
System.out.println("Illustrative rate: " + rate);
```

35. Java star pattern

Nested loops commonly print patterns: the outer loop controls rows and the inner loop controls the number of stars per row.

```
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
// *
// * *
// * * *
// * * * *
// * * * * *
```

5. Agile methodology

36. What is Agile?

Agile is an iterative and incremental way of delivering software. Teams work in short cycles, collaborate with stakeholders, adapt to feedback and deliver usable increments frequently. Scrum is one framework used to organise Agile work; Agile itself is broader than Scrum.

37. When do we go for Agile?

Agile is useful when requirements may evolve, feedback is available regularly, work can be delivered in increments, and the team can collaborate closely. It is not automatically the best fit for every project: strict contractual, regulatory, hardware or dependency constraints may require additional controls or a hybrid approach.

38. What is a retrospective?

A sprint retrospective is a Scrum event at the end of a sprint where the team reflects on collaboration, process and quality, then agrees on actionable improvements. It is not a blame session and differs from the sprint review, which focuses on the product increment and stakeholder feedback.

39. Explain Agile process

A typical Scrum cycle includes product backlog refinement, sprint planning, daily Scrum, design/development/testing during the sprint, sprint review and sprint retrospective. QA contributes from requirement refinement onward, creates test scenarios, tests continuously, reports risks and supports release decisions. The team inspects outcomes and adapts its approach each sprint.

6. Project and behavioural questions

40. Tell me about yourself

Sample: “I have around 13 years of experience in software testing across manual and automation testing. My core automation experience includes Java, Selenium WebDriver, TestNG, Cucumber and a POM-based hybrid framework. I have also worked with Playwright, ETL testing and JMeter. My responsibilities include understanding requirements, preparing test scenarios, developing and maintaining automation, analysing failures, managing defects and collaborating with Agile teams. I focus on reliable automation, maintainability and release quality. I am looking for a role where I can contribute more to automation strategy and QA leadership.” Adjust the tool list and years to match your actual experience.

41. Tell about yourself chronologically

Organise the answer from earliest role to current role: (1) degree or starting point, (2) first testing role and foundational skills, (3) transition into automation and tools learned, (4) important projects and expanded responsibilities, (5) current role, framework and leadership contributions, and (6) why this role is your next step. Keep it concise and focus on career progression, not every job task.

42. Explain your current project

Describe the business purpose and users of the application, your team and responsibilities, technology stack, test approach, automation architecture and current quality challenges. Example structure: “The project supports [business workflow]. I clarify requirements, create test cases, execute functional/regression testing and automate stable workflows using Selenium Java. We use POM to separate page locators/actions from test logic and Cucumber where business-readable scenarios are useful. TestNG, Maven, Git and reporting tools support execution and maintenance.” Use accurate, approved project facts in each example.

43. What is your role in the project?

Explain responsibilities you personally own: requirement analysis, risk assessment, test estimation, test scenario design, manual and automation execution, framework maintenance, defect triage, regression, sprint reporting, mentoring and release support. Distinguish your individual contribution from the team’s work. Add one real example of how your action improved quality or delivery.

44. Challenges faced while automating

Typical challenges include dynamic DOM changes, timing and synchronisation, stale elements, duplicate locators, flaky tests, environment instability, test data dependencies, OTP/MFA, file uploads/downloads and changing requirements. Explain one genuine example using Situation, Task, Action and Result. Describe how you reproduced it, collected evidence, found the root cause, improved waits or locators, and validated the fix.

45. What types of testing do you perform?

A role-appropriate answer may include functional, smoke, sanity, regression, retesting, integration, system, API, database validation, compatibility and exploratory testing. Mention performance or security testing only to the extent you have actually performed it. Explain which types you own and which are performed by specialised teams.

46. How does a bank lending process work?

At a high level, a lender receives an application, verifies identity and documents, assesses income and repayment capacity, checks credit history, evaluates risk and eligibility, approves or rejects the application, communicates terms, completes agreements and disburses funds. Repayment schedules, monitoring, collections and closure follow. Actual steps depend on the lender, product and local regulation. In a QA interview, you could test eligibility rules, interest calculations, mandatory fields, approval workflow, audit trail, access control, notifications and repayment schedule accuracy.

47. What do you know about Agilysys?

Agilysys is a technology company focused on hospitality software and solutions, serving areas such as hotels, resorts, casinos, cruise lines and other hospitality operations. Its product areas have included property management, point of sale, reservations, inventory/procurement and guest-facing experiences. Before an interview, verify the latest company profile and the exact role/product from the official company website and job description. Connect your answer to quality, integrations, payments, reservations or guest workflows only where relevant.

7. Basic logical reasoning and data interpretation

Sample 1: Number series

Question: 2, 4, 8, 16, __. Each number doubles, so the next number is 32.

Sample 2: Percentage

Question: Sales increase from 200 to 250. Increase = 50. Percentage increase = 50 / 200 × 100 = 25%.

Sample 3: Average

Question: Five values have a total of 150. Average = 150 / 5 = 30.

Sample 4: Ratio

Question: The ratio of QA to developers is 2:3 and the team has 25 people. Total parts = 5; each part = 25/5 = 5. QA = 10 and developers = 15.

Sample 5: Probability

Question: A fair die is rolled. Probability of an even number = 3/6 = 1/2 because the even outcomes are 2, 4 and 6.

Data interpretation tip

Read the chart title, labels, units and time period first. Check whether a question asks for a difference, ratio, average or percentage change. Use the correct base for percentage calculations and estimate before computing to catch implausible answers.

Final interview checklist

- Prepare a 60–90 second introduction and a chronological career summary.
- Prepare one clear explanation of your project, framework, POM and personal role.
- Practise OOP, overriding vs overloading, static blocks, strings and collections.
- Practise palindrome, Fibonacci, character frequency, array summing and star patterns.
- Revise SQL GROUP BY, HAVING, ORDER BY, DELETE vs TRUNCATE and basic PL/SQL.
- Know the difference between BVA, equivalence partitioning, test coverage and code coverage.
- Use only real tools, achievements, domains and metrics in your answers.