



1A: Selenium Basics

1

What is Selenium and its components?

Selenium is an open-source toolset used to automate web applications. It has 4 main parts: Selenium IDE (record/playback), RC (server-based tool, now outdated), WebDriver (direct browser control using native APIs), and Grid (run tests across multiple machines in parallel).

2

Difference between Selenium IDE, RC, WebDriver, and Grid

IDE is a simple recording tool. RC needed a server and was slow. WebDriver controls browsers natively without server dependency. Grid allows tests to run on different machines simultaneously for faster execution.

3

What are locators in Selenium?

Locators are methods to identify web elements. Selenium supports ID, Name, ClassName, TagName, LinkText, PartialLinkText, XPath, and CSS Selector to find elements on the page.

4

Difference between findElement() and findElements()

findElement() fetches the first matching element or throws an exception if not found. findElements() returns a list of elements and gives an empty list if no matches are found.

5

What is Page Object Model (POM)?

POM is a design pattern where each page of the application is represented as a class with WebElements and methods, making tests reusable, readable, and easy to maintain.



by **Sreenidhi Rajakrishnan**



1B: Selenium Basics

1

How do you handle dropdowns?

Selenium provides the `Select` class to handle dropdowns. You can select options by visible text, value, or index using methods like `selectByVisibleText()`.

2

What are waits in Selenium?

Waits handle synchronization issues. Implicit Wait sets a global timeout. Explicit Wait waits for specific conditions. Fluent Wait is like Explicit but allows polling frequency and exception handling.

3

Difference between `driver.close()` and `driver.quit()`

`close()` closes only the current browser window. `quit()` closes all windows and ends the WebDriver session completely.

4

What is synchronization in Selenium?

Synchronization aligns Selenium's execution with web application speed by using waits, avoiding `ElementNotFound` errors due to slow loading pages.

5

Can Selenium test desktop apps?

No. Selenium is designed only for web applications. For desktop apps, tools like WinAppDriver, Winium, or Autolt are used.



by Sreenidhi Rajakrishnan



1C: Selenium Basics (Part 3)

1

What is the purpose of Actions class?

The Actions class simulates complex user actions such as mouse hover, drag-and-drop, right-click, and double-click events in Selenium.

2

How do you perform drag and drop?

Using Actions class:

```
Actions action = new Actions(driver);  
action.dragAndDrop(source, target).perform();
```

It clicks the source element and drops it on the target.

3

How do you switch to frames?

You can switch to frames using frame name, frame index, or WebElement reference with `driver.switchTo().frame()` method.

4

How do you handle browser notifications/popups?

Browser notifications can be disabled by setting browser options, like `ChromeOptions().addArguments("--disable-notifications")`.

5

How do you capture screenshots?

Use TakesScreenshot interface:

```
TakesScreenshot ts = (TakesScreenshot) driver;  
File src = ts.getScreenshotAs(OutputType.FILE);
```

This captures the current screen.



by **Sreenidhi Rajakrishnan**



2A: XPath & Web Elements (Part 1)

1

What is XPath?

XPath is a language to locate elements in an XML or HTML document. In Selenium, XPath is heavily used when other locators like ID or Name are not reliable.

2

Difference between absolute and relative XPath

Absolute XPath starts from the root (html/body/div...), prone to break easily. Relative XPath starts anywhere in DOM (//div[@class='menu']), making it more flexible.

3

Use of contains() and starts-with() in XPath

contains() is useful when attribute values are partially dynamic. starts-with() is used when you know the starting pattern of attribute values.

4

Write XPath to find a button with specific text

Example:

```
//button[text()='Submit']
```

It finds the button where exact text is "Submit".

5

How to write dynamic XPath?

Combine functions like contains(), starts-with() and logical operators (and, or) to create XPath that adapts to minor UI changes.



by Sreenidhi Rajakrishnan



2B: XPath & Web Elements (Part 2)

1

What are XPath axes?

XPath axes like parent, child, following-sibling, preceding are used to navigate DOM tree relative to a particular node.

2

Difference between XPath and CSS Selector

XPath can traverse both forward and backward in DOM, while CSS can only move forward (child to descendant). XPath is slower but more powerful.

3

How do you locate hidden elements?

You can use JavaScriptExecutor to manipulate visibility or interact with parent elements containing hidden fields.

4

What is StaleElementReferenceException?

It occurs when the WebElement was available during page load but got detached from DOM later. You must re-locate the element.

5

How do you resolve ElementNotInteractableException?

Wait for element visibility or use JavaScript click if direct interaction fails.



by **Sreenidhi Rajakrishnan**



2C: XPath & Web Elements (Part 3)

1

How to highlight a WebElement using JavaScript?

Example:

```
JavaScriptExecutor js = (JavaScriptExecutor) driver;  
js.executeScript("arguments[0].style.border='3px solid red'",  
element);
```

Highlights the element visually during tests.

2

How do you validate a tooltip?

Fetch the title attribute of the element using `getAttribute("title")` and assert the value.

3

Scroll to element using JavaScriptExecutor

Use:

```
js.executeScript("arguments[0].scrollIntoView(true);", element);
```

Brings the element into the visible area.

4

Capture all links on a page

Find elements with `tagName a` and extract `href` attributes.

5

How to verify broken links?

Send HTTP requests to extracted URLs and check for valid response codes (200 OK expected).



by **Sreenidhi Rajakrishnan**



3A: Selenium Java Coding (Part 1)

1

Login test using POM

Create separate page classes for login screen (LoginPage.java) and invoke methods in your test class to enter username, password, and click login.

2

Click all checkboxes on a page

Use
`driver.findElements(By.xpath("//input[@type='checkbox']")).click()`
inside a loop to click each checkbox.

3

Upload file in Selenium

Directly send the file path to file input:

```
driver.findElement(By.id("upload")).sendKeys("path/to/file.txt");
```

4

Handle multiple windows/tabs

Get all window handles and switch:

```
for(String handle : driver.getWindowHandles()){  
    driver.switchTo().window(handle);  
}
```

5

Handle calendar date pickers

Directly pick the date if possible; otherwise, send JavaScript to set date field value.



by Sreenidhi Rajakrishnan



3B: Selenium Java Coding (Part 2)

1

Capture and save screenshot on failure

Capture screenshot inside a catch block or @AfterMethod when a test fails to help debugging.

2

Retry failed test using listener

Implement IRetryAnalyzer interface and link it to TestNG listeners to auto-retry flaky tests.

3

Handle autocomplete dropdown

Send partial input, wait for dropdown options, and select matching option via click.

4

Extract dynamic table data

Use loops to traverse table rows and columns dynamically using XPath expressions.

5

Hover over an element

Use Actions class:

```
Actions act = new Actions(driver);  
act.moveToElement(element).perform();
```



by **Sreenidhi Rajakrishnan**



3C: Selenium Java Coding (Part 3)

1

Validate all dropdown options

Fetch all options using Select class and verify them against expected values.

2

Highlight a failed element

Change element's CSS border using JavaScriptExecutor to indicate failure visually.

3

Scroll and validate lazy-loaded content

Scroll page by JavaScript or Actions class to trigger loading of dynamic content.

4

Use of soft assertions

SoftAssert allows the test to continue execution even after one assertion fails, collecting all failures at once.

5

Add FluentWait for a WebElement

Create FluentWait with polling interval and ignore exceptions like NoSuchElementException until timeout.



by Sreenidhi Rajakrishnan



4A: Core Java Concepts (Part 1)

1

Difference between List, Set, and Map

List stores ordered elements and allows duplicates. Set stores unique elements with no particular order. Map stores key-value pairs where keys must be unique.

2

HashMap vs Hashtable

HashMap is non-synchronized, faster, and allows one null key and multiple null values. Hashtable is synchronized (thread-safe) but slower and does not allow any null key or value.

3

ArrayList vs LinkedList

ArrayList is better for accessing elements (random access), whereas LinkedList is faster for insertions and deletions due to its node-based structure.

4

What is method overloading?

Method overloading means creating multiple methods in the same class with the same name but different parameters (different type or number).

5

What is method overriding?

Method overriding means redefining a parent class method in a child class with the same method signature to provide a specific implementation.



by **Sreenidhi Rajakrishnan**



4B: Core Java Concepts (Part 2)

1

What are interfaces in Java?

An interface in Java is a blueprint of a class with abstract methods. A class implements an interface to inherit its abstract methods, achieving full abstraction and multiple inheritance.

2

Abstract class vs Interface

Abstract class can have both abstract and concrete methods. Interfaces can only have abstract methods (until Java 8) and support multiple inheritance, unlike abstract classes.

3

What are static blocks?

Static blocks are used to initialize static variables. They are executed once when the class is loaded into memory before the constructor or main method.

4

Difference between final, finally, and finalize

final is used to declare constants or prevent inheritance. finally is a block that executes after a try-catch block, regardless of exceptions. finalize() is a method called by Garbage Collector before object destruction.

5

What is polymorphism?

Polymorphism means the ability of an object to take multiple forms – achieved via method overloading (compile-time polymorphism) and method overriding (runtime polymorphism).



by **Sreenidhi Rajakrishnan**



4C: Core Java Concepts (Part 3)

1

Use of this and super keywords

this refers to the current class instance. super refers to the immediate parent class object and is used to access parent methods or constructors.

2

Exception handling in Java

Java provides try-catch-finally blocks to handle exceptions. This helps prevent program crashes and allows alternative code execution when exceptions occur.

3

What are collections in Java?

Collections are frameworks that provide architecture to store and manipulate groups of objects. Key interfaces include List, Set, and Map.

4

Purpose of wrapper classes

Wrapper classes like Integer, Double, Character wrap primitive data types into objects so they can be used in collections, which require objects.

5

What are lambda expressions?

Lambda expressions provide a concise way to represent functional interfaces, making code easier to read and write in Java 8+. Example:

```
(int a, int b) -> a + b;
```



by **Sreenidhi Rajakrishnan**



5A: Java Coding Exercises (Part 1)

1

Reverse a string

You can reverse a string by converting it to a character array and swapping elements or by using `StringBuilder`'s `reverse()` method. Example:

```
new StringBuilder("hello").reverse().toString();
```

2

Count frequency of characters in a string

Use a `HashMap` to store characters as keys and their counts as values by iterating through the string.

3

Find duplicates in an array

Use a `HashSet`. If `add()` returns false for an element, it's a duplicate.

4

Find second largest number in an array

Traverse the array while maintaining two variables – one for the largest and another for the second largest value.

5

Palindrome check

Compare the original string to its reverse. If both are equal, it is a palindrome.



by Sreenidhi Rajakrishnan



5B: Java Coding Exercises (Part 2)

1 Implement binary search

Binary search works by dividing a sorted array and comparing the mid element to the target. Adjust search range based on comparison.

2 Convert HashMap to List

Use:

```
List keys = new ArrayList<>(hashMap.keySet());  
List values = new ArrayList<>(hashMap.values());
```

to extract keys or values into a list.

3 Swap two numbers without using a third variable

Use arithmetic:

```
a = a + b;  
b = a - b;  
a = a - b;
```

4 Remove duplicates from a list

Convert list to a HashSet and then back to a list:

```
new ArrayList<>(new HashSet<>(list));
```

5 Count vowels and consonants in a string

Loop through the string. If the character is a vowel (a, e, i, o, u), increment vowel count; otherwise, if it's a letter, increment consonant count.



by **Sreenidhi Rajakrishnan**



5C: Java Coding Exercises (Part 3)

1

Factorial using recursion

Example:

```
int factorial(int n){  
    if(n==0) return 1;  
    return n * factorial(n-1);  
}
```

2

Print Fibonacci series

Start with 0 and 1, then print the sum of last two numbers iteratively.

3

Sort a list alphabetically

Use Collections.sort():

```
Collections.sort(list);
```

4

Merge two sorted arrays

Create a new array, compare elements of both input arrays, and insert the smaller one first.

5

Create a custom exception class

Extend the Exception class:

```
class MyException extends Exception {  
    public MyException(String message) {  
        super(message);  
    }  
}
```



by Sreenidhi Rajakrishnan



6A: TestNG Essentials (Part 1)

1

What is TestNG?

TestNG (Test Next Generation) is a powerful testing framework for Java, inspired by JUnit but with additional features like parallel execution, groups, priorities, dependency testing, and flexible reporting.

2

TestNG vs JUnit

TestNG offers more flexible configuration (with XML), parallel test execution, better annotations, built-in dependency management between tests, and custom reporting compared to the basic structure of JUnit.

3

Common TestNG annotations

- `@Test`: Marks a test method.
- `@BeforeMethod`/`@AfterMethod`: Run before and after every test method.
- `@BeforeClass`/`@AfterClass`: Run before and after all test methods in a class.
- `@BeforeSuite`/`@AfterSuite`: Run before and after all tests in the suite.

4

What is a TestNG XML suite file?

TestNG XML lets you define which classes, methods, and groups to execute, configure parallel execution, and set parameters for your tests – all externally without modifying code.

5

How do you group test cases in TestNG?

You use the `groups` attribute inside `@Test`. In XML, you can include or exclude specific groups to run only selected tests.



by **Sreenidhi Rajakrishnan**



6B: TestNG Essentials (Part 2)

1

How to run tests in parallel in TestNG?

Add `parallel="methods"` or `parallel="classes"` attribute in your suite XML and specify thread-count. This runs tests or classes simultaneously.

2

What is a DataProvider in TestNG?

DataProvider is a TestNG feature to supply multiple sets of test data to a single test method, enabling data-driven testing easily.

3

How to skip a test case in TestNG?

Set `enabled=false` inside the `@Test` annotation or throw a `SkipException` during runtime based on a certain condition.

4

SoftAssert vs Assert

Assert stops the test immediately on failure. SoftAssert continues test execution even after an assertion failure, collecting all errors to be reported at the end.

5

How do you use IRetryAnalyzer?

Implement `IRetryAnalyzer` interface to automatically retry failed test cases a certain number of times before marking them as failed.



by Sreenidhi Rajakrishnan



6C: TestNG Essentials (Part 3)

1

Dependency between test methods

Use `dependsOnMethods` in `@Test` to define that one test should run only after another succeeds. If the parent fails, the dependent test is skipped.

2

TestNG Listeners

Listeners are interfaces like `ITestListener`, `ISuiteListener` that help capture test events like success, failure, start, and finish – useful for reporting or screenshots on failure.

3

Execution order control in TestNG

Use the `priority` attribute in `@Test`. Lower priority values run first. If priority isn't specified, methods run alphabetically.

4

Prioritization of test cases

Assign integer priority to tests (`priority=1`, `priority=2`). Tests execute based on ascending priority order, helping in controlled execution.

5

Generating TestNG reports

TestNG automatically creates HTML and XML reports after test execution. These reports provide test results summary, details, and can be customized further.



by **Sreenidhi Rajakrishnan**



7A: Maven Essentials (Part 1)

1

What is Maven and its purpose?

Maven is a build and dependency management tool that automates compiling, testing, packaging, and deploying Java projects, while also managing external libraries efficiently through a simple XML configuration.

2

What is pom.xml?

pom.xml is the core configuration file of Maven projects where dependencies, plugins, build instructions, and project info are declared.

3

What are dependencies in Maven?

Dependencies are external libraries required by a project (like Selenium, TestNG). Maven automatically downloads these from central repositories.

4

Life cycle phases of Maven

- validate: Check project structure.
- compile: Compile source code.
- test: Run unit tests.
- package: Create JAR/WAR.
- verify: Check integration tests.
- install: Install to local repo.
- deploy: Upload to remote repo.

5

Maven commands: clean, install, test

- mvn clean: Deletes target folder.
- mvn install: Builds and installs locally.
- mvn test: Runs unit or automation tests.



by **Sreenidhi Rajakrishnan**



7B: Maven Essentials (Part 2)

1

What is a repository in Maven?

A repository is a place where Maven stores project artifacts (JARs, libraries). Repositories can be local (user machine) or remote (online servers like Maven Central).

2

Local vs Remote Repository

Local repo is on your machine (default: .m2 folder). Remote repo is online like Maven Central. Maven checks local first, then remote.

3

How to add TestNG dependency in Maven?

Add this to pom.xml:

```
org.testng  
testng  
7.4.0
```

4

What is Maven Surefire Plugin?

It runs unit tests automatically during Maven build. Useful for executing TestNG or JUnit test classes.

5

How to skip test execution in Maven?

Use command:

```
mvn install -DskipTests
```

to compile and package code without running tests.



by **Sreenidhi Rajakrishnan**



7C: Maven Essentials (Part 3)

1

What are Maven profiles?

Profiles allow setting different configurations (like dev, QA, production) which can be activated by command-line switches during build.

2

Parent and Child POM concept

A Parent POM defines common configurations. Child modules inherit from the parent, promoting reusability and reducing duplication across projects.

3

Creating a Maven project manually

Run:

```
mvn archetype:generate
```

and follow prompts for project creation like groupId, artifactId.

4

Managing dependency versions via Maven

Declare versions in block in parent POM to ensure uniform versioning across child modules.

5

Maven integration with Jenkins

Jenkins can invoke Maven goals like clean, install, and test during build pipelines for continuous integration automation.



by **Sreenidhi Rajakrishnan**



8A: Cucumber & BDD (Part 1)

1

What is BDD?

BDD (Behavior Driven Development) is a development approach that enhances collaboration between developers, testers, and business stakeholders by writing scenarios in plain English that describe application behavior.

2

What is Cucumber?

Cucumber is an open-source tool that supports BDD. It reads specifications written in Gherkin language and runs corresponding automation scripts behind them.

3

What is Gherkin?

Gherkin is a simple, readable language used to write Cucumber test scenarios. It uses keywords like Given, When, Then, And, But to define steps.

4

Structure of a feature file

A feature file contains:

- Feature: Title of the functionality.
- Scenario: Specific test case.
- Steps: Series of Given-When-Then statements describing actions and expected outcomes.

5

What is a step definition?

A step definition links Gherkin steps to Java methods. When Cucumber encounters a step in the feature file, it calls the matching step definition code.



by Sreenidhi Rajakrishnan



8B: Cucumber & BDD (Part 2)

1

What are hooks in Cucumber?

Hooks (@Before, @After) are blocks of code executed before and after each scenario. They are commonly used for setup (like launching browsers) and teardown (like closing browsers).

2

What is Background in Cucumber?

Background allows you to define common steps that run before each scenario in a feature file, reducing duplication of Given conditions.

3

What is a Scenario Outline?

Scenario Outline lets you run the same scenario multiple times with different sets of data. Data is provided using an Examples table.

4

How to use tags in Cucumber?

Tags (e.g., @SmokeTest) allow grouping of scenarios so you can control which scenarios to execute by including/excluding tags in Runner file.

5

Integration of Cucumber with TestNG

Use the AbstractTestNGCucumberTests class to run Cucumber feature files with TestNG, enabling you to use TestNG's features like parallel execution.



by Sreenidhi Rajakrishnan



8C: Cucumber & BDD (Part 3)

1

Data tables in Cucumber

DataTables are used to pass complex input data (like multiple rows of inputs) to step definitions in a structured form, supporting maps and lists.

2

Generating Cucumber reports

Cucumber can generate basic HTML reports. For advanced reports, integrate plugins like ExtentReports, Cucumber Reporting Plugin, or Allure.

3

Use of Runner class in Cucumber

Runner class, annotated with `@RunWith` and `@CucumberOptions`, configures which features to run, sets paths for feature files and step definitions, and controls report generation.

4

How to reuse steps across scenarios

Define common step definitions in a separate package and reference them from multiple scenarios or feature files to promote reusability.

5

Cucumber options explained

Options like features, glue, tags, plugin, monochrome, and strict control feature path, step binding, report output, and execution behavior.



by Sreenidhi Rajakrishnan



9A: Git & GitHub (Part 1)

1

What is Git?

Git is a distributed version control system that tracks changes to files and allows multiple developers to work together on a project efficiently.

2

Initialize a Git repo

Use:

```
git init
```

to initialize a new local repository.

3

git clone, git pull, git push

- git clone: Copies a remote repo locally.
- git pull: Fetches latest changes and merges.
- git push: Uploads your local commits to the remote repo.

4

Difference between fetch and pull

fetch only downloads new data; it doesn't integrate it. pull fetches and merges changes automatically.

5

What is .gitignore?

A file listing paths to be ignored by Git, preventing unnecessary or sensitive files (like logs, .class files) from being tracked.



by **Sreenidhi Rajakrishnan**



9B: Git & GitHub (Part 2)

1

Creating and switching branches

Use:

```
git checkout -b new-branch-name
```

to create and switch to a new branch in one command.

2

Merging branches

Use:

```
git checkout main  
git merge feature-branch
```

to merge changes from one branch into another.

3

Resolving merge conflicts

Manually edit the conflicting files, mark conflicts as resolved, and commit the changes after reviewing.

4

What is git stash?

Temporarily saves uncommitted changes to clean the working directory without committing them.

5

How to revert a commit

Use:

```
git revert commit_hash
```

to undo changes made by a specific commit without rewriting history.



by **Sreenidhi Rajakrishnan**



9C: Git & GitHub (Part 3)

1

GitHub vs GitLab

GitHub is the most popular cloud-based Git hosting service. GitLab provides similar features but with built-in CI/CD pipelines and more control over private hosting.

2

Pull request vs merge

A pull request is a request to merge code changes into another branch after review. Merging is the actual combining of branch changes.

3

Forking a repo

Forking creates your own copy of someone else's repository on GitHub, allowing you to freely experiment without affecting the original.

4

Setting upstream remote

Use:

```
git remote add upstream original_repo_url
```

to track and fetch changes from the original repository you forked.

5

Tagging releases in Git

Use:

```
git tag -a v1.0 -m "Release version 1.0"
```

to create a versioned snapshot of the project for release.



by **Sreenidhi Rajakrishnan**



10A: Agile & Scrum (Part 1)

1

What is Agile?

Agile is an iterative and incremental software development methodology emphasizing collaboration, flexibility, continuous feedback, and delivering working software in small, frequent releases.

2

Agile vs Waterfall

Waterfall is linear and sequential with late feedback, while Agile is flexible, iterative, and allows continuous customer involvement and feedback at every stage.

3

Scrum roles: PO, SM, Dev Team

- **Product Owner:** Defines product vision and backlog.
- **Scrum Master:** Facilitates scrum practices, removes impediments.
- **Development Team:** Delivers potentially shippable increments every sprint.

4

What is a user story?

A user story describes a feature or requirement from an end-user perspective. It usually follows the format: "As a [user], I want [goal] so that [reason]."

5

Definition of Done

A shared understanding within the team of what must be completed for a user story to be considered finished – includes coding, testing, reviews, and documentation.



by **Sreenidhi Rajakrishnan**



10B: Agile & Scrum (Part 2)

1

What is Sprint Planning?

Sprint Planning is a meeting at the start of a sprint where the team selects and commits to backlog items (user stories) they plan to deliver.

2

What is a Daily Standup?

A 15-minute team meeting held every day of the sprint where team members answer three questions:

- What did you do yesterday?
- What will you do today?
- Any blockers?

3

What is a Sprint Review?

A meeting at the end of the sprint to showcase the completed work to stakeholders and gather feedback for future improvements.

4

What is a Sprint Retrospective?

A meeting after Sprint Review where the team reflects on the sprint process and discusses what went well, what didn't, and how to improve.

5

Product Backlog vs Sprint Backlog

- **Product Backlog:** List of all desired work on the project.
- **Sprint Backlog:** Subset of product backlog items selected for the current sprint.



by **Sreenidhi Rajakrishnan**



10C: Agile & Scrum (Part 3)

1

Story points estimation

Story points estimate the relative complexity or effort of a user story, typically using Fibonacci sequence (1, 2, 3, 5, 8, etc.).

2

Burndown chart

A graphical representation showing the remaining work in the sprint over time, helping the team track progress.

3

Acceptance criteria

Conditions that a user story must satisfy to be accepted as complete by the Product Owner.

4

What is backlog grooming?

Regular sessions where the team refines, prioritizes, estimates, and clarifies items in the product backlog.

5

Velocity tracking

Measurement of the number of story points completed by a team in a sprint, used for future sprint planning and forecasting.



by Sreenidhi Rajakrishnan



11A: STLC & Bug Lifecycle (Part 1)

1

What is STLC?

Software Testing Life Cycle (STLC) defines a sequence of activities conducted during software testing, from requirement analysis to test closure, ensuring structured and systematic testing.

2

Phases in STLC

- Requirement Analysis
- Test Planning
- Test Case Design
- Test Environment Setup
- Test Execution
- Test Cycle Closure

3

Entry and Exit criteria in STLC

Entry criteria are preconditions to start a phase (e.g., approved requirements). Exit criteria define conditions to mark a phase as completed (e.g., 100% test cases executed).

4

What is a test plan?

A test plan is a detailed document describing the test strategy, scope, objectives, resources, schedules, deliverables, and risks related to the testing process.

5

What is a test case?

A test case is a document specifying input, execution conditions, testing procedure, and expected results to validate a particular feature or functionality.



by **Sreenidhi Rajakrishnan**



11B: STLC & Bug Lifecycle (Part 2)

1

What is a test strategy?

A high-level document outlining the testing approach of the organization, covering testing levels, types, tools, metrics, and risk management.

2

What is test data?

Test data consists of input values that are used to execute test cases. Good test data improves the quality and coverage of testing.

3

How do you maintain test artifacts?

Test artifacts like test cases, test scripts, and test results are maintained using version control tools like Git, test management tools like Jira, or stored in repositories.

4

What is defect lifecycle?

Defect lifecycle is the journey of a defect through stages like New → Assigned → Open → Fixed → Retest → Verified → Closed or Reopen.

5

Bug states from discovery to closure

- New
- Assigned
- Open
- Fixed
- Retested
- Verified
- Closed



by Sreenidhi Rajakrishnan



11C: STLC & Bug Lifecycle (Part 3)

1

Severity vs Priority

Severity defines the impact of the bug on functionality. Priority defines how urgently the bug needs to be fixed. High severity does not always mean high priority.

2

Blocker vs Critical bug

A Blocker stops all progress (e.g., application crash). A Critical bug severely impacts functionality but might have a workaround.

3

Duplicate bug handling

Duplicate bugs are marked and closed after verifying that the issue already exists or is being tracked under another bug ID.

4

Defect triage meeting

A meeting where defects are discussed, prioritized, and assigned based on severity, priority, and business impact with stakeholders like Dev, QA, and Product Owners.

5

How do you write a good bug report?

A good bug report includes a clear title, reproducible steps, expected vs actual results, environment details, severity, screenshots/logs, and should be concise and objective.



by **Sreenidhi Rajakrishnan**



12A: Framework Design (Part 1)

1 What is a test automation framework?

It's a set of guidelines, practices, and tools designed to make automation tests efficient, reusable, and maintainable. Examples include POM, Data-Driven, Hybrid frameworks.

2 Types of automation frameworks

- Linear
- Modular
- Data-Driven
- Keyword-Driven
- Hybrid
- BDD Frameworks like Cucumber

3 Folder structure in a Selenium framework

Common folders:

- src/main/java for framework code
- src/test/java for test cases
- resources for configurations
- reports for output
- drivers for WebDriver binaries

4 What is PageFactory in Selenium?

PageFactory helps initialize WebElements using @FindBy annotations. It improves code readability and manages page element initialization effectively.

5 Config file management

Properties files (.properties) or YAML files are used to externalize environment-specific data like URLs, credentials, and timeouts for easier maintenance.



by **Sreenidhi Rajakrishnan**



12B: Framework Design (Part 2)

1

Reusable utility classes

Utilities like WaitHelper, ScreenshotHelper, ExcelReader are created separately to reuse common functionalities across different tests.

2

Data handling via Excel/JSON

Test data is stored in Excel (using Apache POI) or JSON (using Jackson/GSON libraries) to allow data-driven testing.

3

Logging using Log4j

Log4j is used to record important events like test execution status, errors, and debug info, making troubleshooting faster.

4

Exception handling in frameworks

Exceptions like NoSuchElementException are handled gracefully using try-catch blocks and custom recovery strategies to make frameworks stable.

5

Test reports (Extent/Allure)

Extent Reports and Allure Reports offer rich HTML reports with pass/fail status, screenshots on failure, and pie charts for test results visualization.



by **Sreenidhi Rajakrishnan**



12C: Framework Design (Part 3)

1

How to make your framework scalable

By following modularity, reusability, centralized configurations, and clear naming conventions, frameworks can be easily expanded for bigger projects.

2

Version control integration

Integrating frameworks with Git ensures code collaboration, branching, merging, and change tracking among multiple team members.

3

Modularizing your tests

Breaking down large tests into small reusable methods improves maintainability and supports better test data parameterization.

4

Running tests in parallel

Tools like TestNG (parallel methods/classes) and Selenium Grid allow running multiple tests simultaneously, saving execution time.

5

Framework CI/CD readiness

Frameworks should support Jenkins/Bamboo integrations, allowing automated test execution on code commits or scheduled nightly runs.



by **Sreenidhi Rajakrishnan**



13A: Jenkins & CI/CD (Part 1)

1

What is Jenkins?

Jenkins is an open-source automation server used to build, test, and deploy code automatically as part of continuous integration and continuous delivery pipelines.

2

How to set up a Jenkins job?

Create a new job, configure SCM (like Git), define build triggers (poll SCM, schedule builds), and add build steps (like Maven goals or shell commands).

3

Trigger builds using Git

Configure webhook on GitHub or GitLab to automatically trigger Jenkins builds on every push to the repository.

4

Run Maven project in Jenkins

Add mvn clean install or mvn test as a build step in Jenkins to compile code and run automated tests.

5

Generate TestNG reports in Jenkins

Post-build actions allow publishing TestNG result XMLs as readable HTML reports or integrating Extent/Allure reports for enhanced visualization.



13B: Jenkins & CI/CD (Part 2)

1

Email notifications after builds

Use the Jenkins Email Extension Plugin to notify developers of build status via emails – successful, unstable, or failed.

2

Pipeline jobs vs freestyle jobs

Pipeline jobs use code-as-configuration (Jenkinsfile) to define build logic, whereas freestyle jobs are simple UI-configured jobs.

3

Jenkins plugins used in test automation

Common plugins include Maven Integration, Git, Email Extension, Allure Report, HTML Publisher, Build Pipeline Plugin.

4

Scheduling builds (cron syntax)

Jenkins supports CRON expressions like `H 1 * * *` to schedule periodic jobs (e.g., nightly builds).

5

Handling test failures in Jenkins

Configure jobs to mark builds as unstable on test failures and use retry mechanisms or rerun failed test suites.



13C: Jenkins & CI/CD (Part 3)

1

How to archive artifacts

In Post-Build Actions, select "Archive Artifacts" and specify files like test reports or JARs to retain them even after builds are completed.

2

Slack/email integration

Integrate Jenkins with Slack or Email servers to get real-time notifications about build statuses, deployments, or test results.

3

Jenkinsfile basics

Jenkinsfile is a text file that defines the stages and steps of a pipeline using either Declarative or Scripted syntax.

4

Shared libraries in Jenkins

Shared libraries allow sharing reusable Groovy functions across multiple Jenkins pipelines, improving maintainability.

5

Running parallel jobs

Jenkins pipeline allows stages to run in parallel using parallel blocks, speeding up build and test execution.



by Sreenidhi Rajakrishnan



14A: REST API & Rest Assured (Part 1)

1

What is an API?

An API (Application Programming Interface) allows two software systems to communicate with each other, exposing data and functionality without sharing internal details.

2

What is REST?

REST (Representational State Transfer) is an architectural style for APIs that uses HTTP methods like GET, POST, PUT, DELETE to perform CRUD operations on resources.

3

What is Rest Assured?

Rest Assured is a popular Java library used to test RESTful APIs. It simplifies sending HTTP requests, validating responses, and automating API testing easily without extensive boilerplate code.

4

Common HTTP methods used

- GET: Retrieve data
- POST: Create new data
- PUT: Update existing data
- DELETE: Remove data

5

How to validate status code with Rest Assured

Example:

```
given().when().get("/users").then().statusCode(200);
```

It checks that the API response has a status code of 200 (OK).



by **Sreenidhi Rajakrishnan**



14B: REST API & Rest Assured (Part 2)

1

How to send query parameters

Use `queryParams()` method to send key-value pairs in the request URL:

```
given().queryParams("page",2).when().get("/users");
```

2

How to send headers in API requests

Use `header()` or `headers()` method to set required request headers like Authorization, Content-Type, etc.

3

Validating JSON response body

Use `body()` method:

```
given().when().get("/users").then().body("data.id", equalTo(1));
```

to validate response fields.

4

Handling authentication in Rest Assured

You can set Basic Auth using:

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

for secured APIs.

5

How to log request and response

Use `log().all()` method to print complete request and response data for easier debugging.



by **Sreenidhi Rajakrishnan**



14C: REST API & Rest Assured (Part 3)

1

Sending POST request with body

Example:

```
given().body(jsonBody).post("/users");
```

where jsonBody is a String or JSON object payload.

2

Path parameters in Rest Assured

Use pathParam() to replace parts of the URL dynamically:

```
given().pathParam("userId",2).get("/users/{userId}");
```

3

What is JSONPath?

JSONPath is a query language to extract parts of a JSON document easily, similar to XPath for XML.

4

Serialization and Deserialization

Serialization: Java object → JSON Deserialization: JSON → Java object (using libraries like Jackson or Gson with Rest Assured).

5

How to validate response headers

Example:

```
given().when().get("/users").then().header("Content-Type","application/json; charset=utf-8");
```



by **Sreenidhi Rajakrishnan**



15A: Real World QA Questions (Part 1)

1

Tell me about your recent testing project

Describe the domain, your role (e.g., Automation Engineer), the tech stack (Selenium, Rest Assured, TestNG), and your key contributions like framework development, CI/CD integration, and handling critical defects.

2

What was the most challenging bug you found?

Mention a complex, hard-to-reproduce bug, how you found it (e.g., through exploratory or edge case testing), and how it impacted the system.

3

How do you convince a developer that it is a bug?

Present reproducible steps, logs, screenshots, and how the behavior deviates from the requirement/user story acceptance criteria objectively and collaboratively.

4

How do you prioritize test cases for execution?

Based on risk, business criticality, defect-prone modules, customer usage frequency, and recent code changes (risk-based testing).

5

How do you perform API vs UI testing?

API testing is faster, stable, and validates business logic directly. UI testing covers user workflows but is slower and more fragile. Both complement each other.



by Sreenidhi Rajakrishnan



15B: Real World QA Questions (Part 2)

1

How do you handle unstable tests in automation?

Root cause analysis, fixing timing issues with explicit waits, avoiding hardcoding, stabilizing test data, and isolating environment dependencies.

2

How do you review a test case?

Check for clarity, completeness, traceability to requirements, correct positive/negative scenarios, and minimal redundancy.

3

How do you handle dynamic elements in Selenium?

Use dynamic XPath with contains(), starts-with(), JavaScriptExecutor if necessary, or wait for elements dynamically.

4

What is risk-based testing?

Testing focused on features with the highest business impact or most prone to failure, maximizing coverage with limited resources.

5

How do you contribute during sprint planning?

Analyze user stories, estimate testing effort, identify potential risks early, ask clarifying questions, and ensure acceptance criteria are clear.



by Sreenidhi Rajakrishnan



15C: Real World QA Questions (Part 3)

1

When do you automate a test case?

Automate when test cases are stable, repetitive, regression-critical, and high-volume, ensuring ROI justifies the automation effort.

2

How do you manage flaky tests?

Regular monitoring, logging root causes, improving synchronization, isolating environment issues, and stabilizing data sources.

3

Key qualities of a good QA Engineer

Attention to detail, strong analytical thinking, curiosity, excellent communication, technical skills in automation and tools, proactive mindset.

4

Metrics you track in test automation

Pass/fail rates, test coverage, defect leakage rate, execution time trends, flaky test rates, and CI build health metrics.

5

Best practices in QA automation

Follow POM, use waits properly, data-driven approach, maintain modular code, log meaningfully, integrate with CI/CD, and ensure proper reporting.



by Sreenidhi Rajakrishnan



16A: Selenium Advanced (Part 1)

1

How do you handle SSL certificate errors in Selenium?

Set ChromeOptions or FirefoxProfile to accept insecure certificates:

```
ChromeOptions options = new ChromeOptions();  
options.setAcceptInsecureCerts(true);
```

2

How do you handle Ajax calls in Selenium?

Use Explicit Waits with conditions like `elementToBeClickable()` or `visibilityOfElementLocated()` to wait for Ajax-loaded elements.

3

How do you capture dynamic popups?

Identify popup locators dynamically, use `WebDriverWait` to handle unpredictable popups, and sometimes use window handles if it's a new window.

4

How do you manage cookies in Selenium?

You can add, delete, or retrieve cookies using `WebDriver`'s `manage().getCookies()`, `addCookie()`, and `deleteAllCookies()` methods.

5

How do you maximize browser window?

Simply use:

```
driver.manage().window().maximize();
```

to maximize the browser window after launching it.



by **Sreenidhi Rajakrishnan**



16B: Selenium Advanced (Part 2)

1

Handling multiple frames

Switch into the frame first, interact with elements, and switch back to the main page:

```
driver.switchTo().frame("frameName");  
driver.switchTo().defaultContent();
```

2

How to work with alerts/pop-ups

Use Alert interface:

```
Alert alert = driver.switchTo().alert();  
alert.accept(); // or alert.dismiss();
```

3

How to handle file download in Selenium?

Use browser profile settings to auto-download files without popups, or integrate with tools like Robot class for download dialogs.

4

How to run Selenium tests headlessly

Set ChromeOptions or FirefoxOptions with headless argument:

```
options.addArguments("--headless");
```

to run tests without UI.

5

How do you capture browser logs in Selenium?

For ChromeDriver:

```
LogEntries logs = driver.manage().logs().get(LogType.BROWSER);
```

useful for finding console errors during tests.



by **Sreenidhi Rajakrishnan**



16C: Selenium Advanced (Part 3)

1

Differences between XPath and CSS

XPath supports traversal both upwards and downwards and is more powerful. CSS selectors are faster and simpler but can only traverse down the DOM tree.

2

How to handle stale element reference exception

Re-locate the element again after the page reloads or dynamic updates happen.

3

Shadow DOM handling in Selenium

Selenium WebDriver cannot directly access shadow DOM. You need to use JavaScriptExecutor to penetrate shadow roots.

4

Capturing network traffic with Selenium

Integrate BrowserMob Proxy or use DevTools Protocol (Chrome DevTools) to capture network logs during test execution.

5

How to rerun failed tests automatically

Implement TestNG's IRetryAnalyzer or configure rerun policies in CI pipelines like Jenkins to rerun flaky tests.



by **Sreenidhi Rajakrishnan**



17A: Agile + Testing (Part 1)

1

Difference between Test Plan and Test Strategy

Test Plan is project-specific and details scope, timelines, resources. Test Strategy is organizational and describes the overall testing approach across projects.

2

How do you participate in Agile ceremonies?

By attending Sprint Planning, Daily Standups, Sprint Reviews, and Retrospectives, contributing to estimations, risk identification, and sprint goals.

3

Importance of retrospectives

Retrospectives provide a safe space for teams to reflect on what worked, what didn't, and continuously improve sprint-over-sprint.

4

Agile Testing Quadrants

Divides testing into four quadrants:

- Unit and component tests
- Functional tests
- System-level tests
- User acceptance tests

5

Difference between Definition of Done and Acceptance Criteria

Definition of Done is a checklist applicable to all stories; Acceptance Criteria are specific conditions that a story must fulfill to be considered complete.



by Sreenidhi Rajakrishnan



17B: Agile + Testing (Part 2)

1

How do you estimate testing tasks in Agile?

Use story points (effort-based) or time-based estimation (hours), considering complexity, dependencies, and risk factors.

2

Pair Testing

Two testers or a tester and developer work together at a single machine to explore and test the application collaboratively.

3

Exploratory Testing

Simultaneously learning, designing, and executing tests without predefined scripts, often uncovering bugs missed in formal testing.

4

How to handle changing requirements

Maintain flexibility, adopt minimal documentation with traceability, communicate changes clearly, and update test cases/scenarios dynamically.

5

QA role during backlog grooming

Review upcoming stories, clarify acceptance criteria, provide testability feedback, and identify testing dependencies early.



by Sreenidhi Rajakrishnan



17C: Agile + Testing (Part 3)

1

What is TDD?

Test-Driven Development is a software development practice where you write unit tests first, then code to make the tests pass.

2

What is BDD vs TDD?

BDD focuses on user behavior and collaboration (Given-When-Then), while TDD focuses on technical correctness via unit tests.

3

How to ensure maximum coverage in Agile

Continuously test throughout the sprint, combine API/UI/DB tests, exploratory testing, and involve non-functional testing early.

4

What is Shift-Left Testing?

Moving testing activities earlier into the software development life cycle (at the requirements, design phase) to catch defects sooner.

5

Test Pyramid concept

Pyramid suggests more unit tests, fewer integration tests, and even fewer UI tests for faster, reliable automation.



by **Sreenidhi Rajakrishnan**



18A: Coding Questions

1

Find missing number in an array

Sum expected numbers ($n*(n+1)/2$) and subtract actual sum from the array elements to find the missing one.

2

Check if two strings are anagrams

Sort both strings and compare them, or use a character frequency map.

3

Find duplicate elements in an array

Use HashSet: if add() fails, the element is a duplicate.

4

Reverse words in a sentence

Split sentence by spaces, reverse the array, and join it back.

5

Find first non-repeating character

Use LinkedHashMap to maintain insertion order and frequency; first entry with frequency 1 is the answer.



by Sreenidhi Rajakrishnan



18B: Coding Questions (Part 2)

1

Check if a string is a palindrome

Compare the original string with its reverse. If both match, it's a palindrome. You can reverse using `StringBuilder`.

2

Print Fibonacci series up to N terms

Start with 0 and 1, then add the last two numbers to generate the next one iteratively.

3

Find the largest element in an array

Initialize a variable with the first element and iterate through the array updating it whenever a larger element is found.

4

Count vowels and consonants in a string

Iterate each character, check if it's a vowel (a, e, i, o, u); otherwise, if it's an alphabet, count as consonant.

5

Find the factorial of a number

Use recursion or a simple loop multiplying numbers from 1 to n.



by **Sreenidhi Rajakrishnan**



18C: Coding Questions (Part 3)

1

Find second highest number in array

Maintain two variables – first highest and second highest – while traversing through the array.

2

Remove duplicates from a string

Use a LinkedHashSet to preserve order and remove duplicates or use a StringBuilder with a seen set.

3

Swap two numbers without a third variable

Use arithmetic operations:

```
a = a + b;  
b = a - b;  
a = a - b;
```

4

Reverse an integer

Extract digits one by one using modulo and division operations, and reconstruct the number.

5

Check Armstrong number

Sum of cubes of each digit equals the number itself (e.g., $153 = 1^3 + 5^3 + 3^3$).



by Sreenidhi Rajakrishnan



19A: Real-World Testing Scenarios (Part 1)

1

How do you test a login page?

Test positive login, invalid credentials, blank fields, SQL injection, password masking, remember me functionality, and session expiration.

2

How do you test a search functionality?

Validate keyword search, partial match, no results scenario, pagination, filtering, and special character handling.

3

How do you test file upload feature?

Test uploading valid files, invalid files, large size files, empty uploads, and interruption during upload.

4

How do you test a shopping cart?

Add multiple items, update quantity, remove items, price recalculation, apply promo codes, check stock validation.

5

How do you test an API without UI?

Use tools like Postman, Rest Assured, or CURL commands to send HTTP requests and validate responses directly.



by Sreenidhi Rajakrishnan



19B: Real-World Testing Scenarios (Part 2)

1

How do you test payment gateway integration?

Validate successful payment, declined transactions, expired card handling, incorrect OTP, network failures, and duplicate payment prevention.

2

How do you test logout functionality?

Ensure session is cleared, restricted access to pages after logout, auto logout on inactivity, and multiple login instances handling.

3

How do you test multi-language support?

Validate UI elements translation, input fields, backend data handling, currency symbols, date/time formats across different languages.

4

How do you test mobile responsiveness?

Test UI/UX on different screen sizes using device emulators, browser developer tools, and verify adaptive layouts.

5

How do you test accessibility features?

Test with screen readers, validate keyboard-only navigation, color contrast compliance, and ARIA roles for assistive technologies.



by **Sreenidhi Rajakrishnan**



19C: Real-World Testing Scenarios (Part 3)

1

How do you test email notifications?

Validate email content, correct recipients, subject line, attachments, and that no duplicate emails are triggered.

2

How do you test user role-based access control?

Ensure users can access only allowed functionalities and restricted content is inaccessible based on role.

3

How do you test load/performance?

Use tools like JMeter or Locust to simulate multiple users and validate system response time, throughput, and server stability under load.

4

How do you test scheduled jobs or cron tasks?

Manually trigger jobs if possible, validate expected execution time, output data generation, and error handling.

5

How do you test version upgrades or deployments?

Validate backward compatibility, migration scripts, configuration changes, rollback procedures, and smoke tests post-deployment.



by Sreenidhi Rajakrishnan



20A: Final QA Behavioral Questions (Part 1)

1

How do you handle pressure situations in testing?

Prioritize tasks based on impact, communicate risks early, stay calm, focus on critical areas first, and escalate when necessary.

2

How do you deal with developers who disagree on a bug?

Stick to facts – show reproducible steps, screenshots, logs, and refer to requirement documents to justify the defect objectively.

3

How do you upskill yourself in testing?

Regularly take online courses, attend webinars, practice coding challenges, read testing blogs, and experiment with new tools.

4

How do you ensure quality in Agile projects?

Participate early in grooming, review user stories critically, maintain strong regression suites, and integrate automation in sprints.

5

How do you ensure a bug is truly fixed?

Retest the defect scenario, perform exploratory testing around impacted areas, and validate against acceptance criteria.



by Sreenidhi Rajakrishnan



20B: Final QA Behavioral Questions (Part 2)

1

How do you handle incomplete requirements?

Proactively seek clarifications from BA/PO, document assumptions, communicate risks, and collaborate closely with stakeholders.

2

How do you document your test results?

Maintain detailed test execution reports with pass/fail status, attach screenshots for failures, log defects, and summarize findings concisely.

3

How do you mentor junior testers?

Conduct code reviews, explain best practices, share learning resources, pair for exploratory sessions, and provide constructive feedback.

4

How do you contribute to process improvements?

Identify repetitive tasks for automation, suggest better tools, recommend smarter workflows, and share retrospective action items.

5

How do you approach testing when you have minimal documentation?

Use exploratory testing, leverage domain knowledge, ask clarifying questions, and create ad-hoc test scenarios on the fly.



by Sreenidhi Rajakrishnan



20C: Final QA Behavioral Questions (Part 3)

1

How do you estimate automation effort?

Analyze complexity of test cases, identify reusable components, assess environment setup needs, and buffer for maintenance/debugging.

2

How do you plan test coverage?

Align coverage with requirements, prioritize high-risk areas, include positive and negative paths, and trace back to acceptance criteria.

3

What is your biggest strength as a QA?

Attention to detail, critical thinking, automation proficiency, proactive problem-solving, and effective communication across teams.

4

How do you handle deadlines in testing?

Focus on critical functionality first, manage scope carefully, escalate risks early, automate regression, and ensure continuous communication.

5

Why should we hire you for this QA Automation role?

I bring strong technical skills in Selenium, Java, TestNG, API Testing, Agile understanding, a passion for quality, and a collaborative mindset to enhance project success.



by Sreenidhi Rajakrishnan