Mock Interview Guide for SDET/QE Roles

Section 1: Automation and Framework Design

1. What is your approach when starting automation for a new project?

Answer (STAR):

- **Situation**: When I joined my previous team, the automation framework was already in place, but there was no clear structure for new modules.
- **Task**: My task was to enhance coverage for a new set of regression tests while keeping code maintainable.
- **Action**: I started by reviewing existing utilities, then created a new Page Object structure for the modules I was responsible for. I ensured reusability by keeping locators and test logic separate, and added proper logging and reporting using TestNG.
- **Result**: The new structure made it easier for new team members to write tests faster and reduced maintenance effort significantly.

---

2. Can you explain the Page Object Model (POM) and why it's important?

Answer:
POM is a design pattern that separates the test logic from the UI locators and actions. Each page in the application has a corresponding class where all element locators and methods are defined.
This keeps tests clean, readable, and easier to maintain. If a locator changes, I only update it in one place. It also promotes reusability across multiple test cases.

---

3. What are the key components you would include in an automation framework?

Answer:

- **Base Class** for driver setup and teardown
- **Page Classes** for each web page
- **Utilities** for reusable functions (like waits, screenshots, or data readers)
- **Test Data Management** with either Excel, JSON, or property files
- **Reporting** through TestNG reports or ExtentReports
- **CI/CD Integration** for running tests automatically via Jenkins or GitHub Actions

---

4. How do you handle dynamic elements in Selenium?

Answer:
I usually handle dynamic elements using **XPath functions** like `contains()` or `starts-with()`. For example, if the ID changes dynamically, I identify a stable attribute pattern.
In cases where timing is an issue, I rely on **explicit waits** with `WebDriverWait` to ensure elements are present before interacting with them.

---

5. How do you decide what to automate?

Answer:
I focus on **high-risk, repetitive, and stable** functionalities. Anything that's frequently tested or has minimal UI volatility is a good candidate. I avoid automating test cases that depend heavily on frequently changing UI elements or require complex data setup that outweighs the benefit.

---

6. Can you explain how you'd integrate your tests into CI/CD?

Answer:
I use **Maven** to manage dependencies and **GitHub Actions** or **Jenkins** for automation runs. Once a code push happens, a job is triggered that installs dependencies, executes test cases in headless mode, and publishes reports. This ensures fast feedback for every commit.

Section 2: Programming and Java Concepts

7. Explain the difference between an interface and an abstract class in Java.

Answer:
An **interface** defines a contract. It has abstract methods without implementation. An **abstract class** can have both abstract and concrete methods.
I use interfaces when I want multiple classes to follow the same method signatures, and abstract classes when I want to share common code among subclasses.

---

8. How do you handle exceptions in automation?

Answer:
I use **try-catch blocks** around code that might fail, like clicking or typing actions. In the catch block, I log the exception and take a screenshot.
This helps me analyze test failures quickly without stopping the full suite execution.

---

9. What are Java Collections, and how do you use them in automation?

Answer:
Collections like **Lists, Sets, and Maps** are used to store and manage data efficiently.
For example, I use Lists to store multiple web elements, Maps to hold key-value data for test input, and Sets when I need unique values during validation.

---

10. What is synchronization, and why is it important in Selenium?

Answer:
Synchronization ensures the script waits for the web page or element to be ready before interacting with it. I mostly use **explicit waits** with conditions like `visibilityOfElementLocated` to avoid flaky tests caused by timing issues.

---

11. What's the difference between `findElement` and `findElements`?

Answer:
`findElement` returns a single WebElement and throws an exception if not found.

`findElements` returns a List of WebElements and returns an empty list if no match is found.
I use `findElements` when validating multiple items, like verifying a list of search results.

---

12. How do you parameterize tests in TestNG?

Answer:
I use the **@DataProvider** annotation in TestNG to supply different sets of data to the same test. This lets me test the same functionality with multiple inputs without duplicating code.

Section 3: Testing Methodologies and Tools

13. How do you handle test data management in automation?

Answer:
I store test data in **JSON or Excel files** and read them dynamically. This makes it easy to update or expand test coverage without touching the test scripts.
In one project, I used Apache POI to read Excel sheets and load them into my tests at runtime.

---

14. What is regression testing, and how do you manage it efficiently?

Answer:
Regression testing ensures that new changes haven't broken existing functionality.
To manage it efficiently, I maintain a **regression suite** that runs automatically after every deployment. I also prioritize tests based on critical business flows.

---

15. How do you ensure test coverage is adequate?

Answer:
I use **traceability matrices** to map test cases back to requirements. This ensures that every user story has both manual and automated coverage. I also collaborate with developers to understand new feature changes early.

---

16. What's the difference between verification and validation?

Answer:

- **Verification** checks whether we built the product right (reviews, static checks).
- **Validation** checks whether we built the right product (functional testing).
  In QA, I focus on both by combining code reviews and test execution.

---

17. How do you test APIs in your projects?

Answer:
I use tools like **Postman** or frameworks like **RestAssured**. I verify status codes, response bodies, and headers, and ensure that data integrity is maintained.
For automation, I integrate API tests within the same framework as UI tests to get end-to-end validation.

---

18. How do you handle flaky tests?

Answer:
I analyze the cause. Whether it's environment timing, data dependency, or locator instability.
Then I fix waits, improve data setup, or refactor locators.
If the environment is unstable, I add retry logic for specific tests using TestNG's retry analyzer.

Section 4: Behavioral / STAR Questions

19. Tell me about a time you improved an existing process.

Answer (STAR):

- **Situation**: Our QE team had frequent test failures due to hard-coded waits.
- **Task**: I wanted to stabilize our automation suite.
- **Action**: I replaced all static waits with explicit waits using ExpectedConditions and added custom wait utilities.

- **Result**: Test execution time decreased by 30%, and flaky failures were almost eliminated.

---

20. Tell me about a challenging bug you found.

Answer (STAR):

- **Situation**: During regression, a checkout test intermittently failed without visible UI errors.
- **Task**: My goal was to identify the root cause.
- **Action**: I used network logs and browser console output to trace the issue. It turned out the backend API occasionally returned a null response.
- **Result**: Developers fixed the API handling logic, and the bug no longer occurred.

---

21. Describe a time you had to work with developers to resolve a defect.

Answer (STAR):

- **Situation**: I noticed that our login feature intermittently failed due to session timeout issues.
- **Task**: I needed to reproduce it consistently.
- **Action**: I collaborated with developers to add debug logs and monitored backend session tokens.
- **Result**: We found a timing mismatch in token refresh, and the fix prevented production login failures.

---

22. How do you prioritize your test cases under time pressure?

Answer:
I use **risk-based testing**. Prioritizing features that have the highest business impact or are most

frequently used by users. I focus on those first, ensuring core functionality works before less critical paths.

---

23. How do you handle situations where you disagree with a developer or product owner?

Answer (STAR):

- **Situation**: Once, a developer felt a reported issue was minor and didn't need fixing.
- **Task**: I believed it affected user experience.
- **Action**: I demonstrated the bug's impact using screenshots and logs to show how it could confuse users.
- **Result**: The developer agreed, fixed it, and later thanked me when customers noticed smoother performance.

---

24. What's your biggest strength as a QE?

Answer:
My biggest strength is bridging the gap between development and testing. Because I started as a developer, I write maintainable, modular test code and understand the logic behind backend changes, which helps in early defect detection.

---

25. What are your career goals in the next year?

Answer:
My short-term goal is to secure an SDET role where I can apply my automation and development experience. In the next year, I plan to strengthen my framework design skills, expand into API and CI/CD test automation, and continuously refine my technical portfolio on GitHub and LinkedIn.