- ☐ **What is TestNG, and how is it different from JUnit?**

  TestNG is a testing framework inspired by JUnit, but it offers additional features like parallel test execution, flexible test grouping, and better reporting.

- ☐ **How do you annotate a method as a TestNG test method?**

  Use the `@Test` annotation, like this:

```
@Test
public void myTestMethod() {
    // Test code here
}
```

- ☐ **What is the purpose of the `@BeforeTest` and `@AfterTest` annotations?**

  `@BeforeTest` is used to run setup code before any test method in the test class, and `@AfterTest` is used to run cleanup code after all test methods in the class.

- ☐ **How do you perform parameterized testing in TestNG?**

  Use the `@DataProvider` annotation to supply data to a test method, like this:
  java

```
@DataProvider(name = "myData")
public Object[][] testData() {
    return new Object[][] { { 1, 2 }, { 3, 4 } };
}

@Test(dataProvider = "myData")
public void myTestMethod(int a, int b) {
    // Test code here using a and b
}
```

- ☐ **Explain how to enable parallel test execution in TestNG.**

  Use the `parallel` attribute in the `<suite>` tag of your testng.xml file or annotate your test class with `@Listeners({ParallelListener.class})` to enable parallel execution.

- ☐ **What is the purpose of the `dependsOnMethods` attribute in the `@Test` annotation?**

  It specifies that a test method depends on the successful execution of one or more other test methods before it can run.

- ☐ **How can you prioritize test methods in TestNG?**

  Use the `priority` attribute in the `@Test` annotation to specify the execution order, where lower values execute first.

- ☐ **What is the purpose of the `@Parameters` annotation?**

  `@Parameters` is used to specify parameters for a test method, and you can define these parameters in your testng.xml file.

- ☐ **How do you perform group testing in TestNG?**

  Use the `groups` attribute in the `@Test` annotation to assign a test method to one or more groups, then include or exclude groups in your testng.xml file.

- ☐ **How can you configure TestNG to run tests in a specific order?**

  You can use the `preserveOrder` attribute in the `<suite>` tag of your testng.xml file to specify that test methods should run in their declared order.

- ☐ **Explain the purpose of the `@Listeners` annotation in TestNG.**

  `@Listeners` is used to add custom listeners to your test class, which can perform actions before or after test methods.

- ☐ **What is Soft Assert in TestNG? Provide an example.**

  Soft Assert allows you to continue executing test steps even after an assertion fails.
  Here's an example:
  java

```
SoftAssert softAssert = new SoftAssert();
softAssert.assertEquals(actual, expected);
// Continue with test steps
softAssert.assertAll(); // This will report all failures at the end
```

- ☐ **Explain TestNG listeners and provide an example of a custom listener.**

TestNG listeners are interfaces that allow you to customize test execution. Here's an example of a custom listener:
java

```
public class MyListener implements ITestListener {
    // Override listener methods like onTestStart, onTestSuccess,
etc.
}
```

- [ ] **How can you skip a test method in TestNG?**

  Use the `@Test(enabled = false)` annotation or the `@Test` annotation with `enabled = true` or `false` to skip or execute a test method.

- [ ] **What is the purpose of the `@DataProvider` and `@Factory` annotations?**

  `@DataProvider` supplies data to test methods, while `@Factory` creates instances of test classes, allowing dynamic test creation.

- [ ] **Explain TestNG's reporting capabilities and how to generate test reports.**

  TestNG provides built-in HTML reports. You can also integrate it with tools like ExtentReports or TestNG's `IReporter` interface for custom reporting.

- [ ] **How can you set up TestNG in a Maven project?**

  Add the TestNG dependency in your pom.xml, and then configure your test classes and suites in a testng.xml file.

- [ ] **What is the purpose of the `@BeforeSuite` and `@AfterSuite` annotations?**

  `@BeforeSuite` runs setup code before all test methods in a suite, and `@AfterSuite` runs cleanup code after all test methods.

- [ ] **Explain how to pass parameters to a TestNG test using the testng.xml file.**

  Define parameters in the `<parameter>` tag inside `<test>` or `<suite>` tags in testng.xml, and reference them using `@Parameters` in your test class.

- [ ] **How can you run TestNG tests from the command line?**

  Use the `java -cp` command with the `org.testng.TestNG` class and specify the testng.xml file as an argument.

- **Execute test cases from class payment?//payment is class name**

```
<suite name="API Automation Smoke Suite">
   <test name="Automation Test Cases">
         <classes>
                        <class name="<nameofthepackage>.payment"/>
         </classes>
  </test>
</suite>
```

- **Execute multiple classes from a package using testng xml suite?**

```
<suite name="API Automation Smoke Suite">
   <test name="Automation Test Cases">
         <classes>
                        <class name="apiautomationeleven.test"/>
                        <class name="apiautomationeleven.registration"/>
                        <class name="apiautomationeleven.payment"/>
         </classes>
  </test>
</suite>
```

- **Explain testNG annotation in sequence with real time scenario?example of @After@Before in TestNG?selenium interview qus/testng**

  Click Here For Answer

- **Execute all classes from a pckage using testng.xml suite?**

```
<suite name="API Automation Smoke Suite">
   <test name="Automation Test Cases">
         <packages>
         <package name="apiAutomationeleven" />
         </packages>
  </test>
</suite>
```

- **How to run automation suites with groups?**

```
<suite name="API Automation Smoke Suite">
   <test name="Automation Test Cases">
         <groups>
         <run>
         <exclude name="brokenTests"  />
```

```
        <include name="SmokeSuite"  />
        </run>
        </groups>
  </test>
</suite>
```

- **How to use dependsOnGroups?**

```
@Test(description="B_Users:Validate 200 status code for /users GET
API",dependsOnGroups="Auth_OAUTH")
```

- **How you can specify your group dependencies in the testng.xml file. You use the <dependencies> tag to achieve this**

```
<suite name="API Automation Smoke Suite">
   <test name="Automation Test Cases">
        <dependencies>
        <group name="B_User" depends-on="Auth_Oauth" />
        </dependencies>
  </test>
</suite>
```

- **How to use @Parameters in TestNG?**

```
@Parameters({ "first-name" })
@Test
public void testSingleString(String firstName) {
  System.out.println("Invoked testString " + firstName);
  assert "Cedric".equals(firstName);
}
```

SCENARIO BASED REAL TIME

- **Question:** Explain how you can achieve parallel test execution in TestNG, and what are the key attributes for parallel configuration?

  Answer: Parallel execution in TestNG can be achieved using the `parallel` attribute at the suite, test, or method level. Key attributes include `methods`, `tests`, `instances`, and `classes`.

- **Question:** How can you implement data-driven testing in TestNG, and what annotations or attributes are involved?

  Answer: Data-driven testing is achieved using the `@DataProvider` annotation and associating it with the `dataProvider` attribute in the `@Test` annotation.

- **Question:** Describe how you handle test dependencies in TestNG, and what annotations are used for this purpose.

  Answer: TestNG allows dependency management using the `dependsOnMethods` and `dependsOnGroups` attributes within the `@Test` annotation.

- **Question:** How do you group tests in TestNG, and what mechanisms are available to control the execution of specific groups?

  Answer: Tests can be grouped using the `@Test(groups = "group_name")` annotation. Execution control involves including or excluding specific groups in the XML suite file.

- **Question:** Explain the role of listeners in TestNG, and how you can generate detailed test reports using listeners.

  Answer: Listeners like `IInvokedMethodListener` and `ITestListener` allow you to customize test behavior and generate detailed reports. Utilize tools like Extent Reports for enhanced reporting.

- **Question:** How can you parameterize your tests in TestNG, and what is the significance of the `@Parameters` annotation?

  Answer: Test parameterization is achieved using the `@Parameters` annotation in conjunction with the `parameter` attribute in the `@Test` annotation.

- **Question:** What is a TestNG suite, and how can you configure and run multiple test classes using XML configuration?

  Answer: A TestNG suite is a collection of test classes. Configuration is done using the XML suite file where you define test classes, groups, and parameters.

- **Question:** How can you handle expected exceptions in TestNG, and what annotations are involved?

  Answer: Use the `@Test(expectedExceptions = Exception.class)` annotation to handle expected exceptions during test execution.

- **Question:** Explain the concept of soft assertions in TestNG and how they differ from traditional assertions.

  Answer: Soft assertions, provided by the `SoftAssert` class, allow the execution of subsequent test steps even if an assertion fails, providing a comprehensive test report.

- **Question:** Discuss how you integrate TestNG with Selenium for automated testing, and what are the benefits of this integration.

  Answer: TestNG and Selenium integration involves creating test scripts using TestNG annotations, allowing better test organization, parallel execution, and reporting.