**1) What is Selenium Webdriver?**

Selenium WebDriver is a collection of open source APIs which are used to automate the testing of a web application. Selenium WebDriver tool is used to automate web application testing to verify that it works as expected. It supports many browsers such as Firefox, Chrome, IE, and Safari.

**2) What is Selenium Grid and when do we go for it?**

* Selenium Grid is a feature in Selenium that allows us to run test cases in different machines across different platforms. The control of triggering the test cases is on the local machine, and when the test cases are triggered, they are automatically executed by the remote machine.

We use Selenium Grid in the following scenarios:

– Execute your test on different operating systems

-Execute your tests on different versions of same browser

-Execute your tests on multiple browsers

-Execute your tests in parallel and multiple threads

**3) What are the advantages of Selenium Grid?**

Below are the benefits of Selenium Grid:

– Selenium Grid gives the flexibility to distribute our test cases for execution.

-Reduces batch processing time.

-Can perform multi-browser testing.

-Can perform multi-OS testing.

**4) What is a Hub in Selenium Grid?**

Hub is the central point to the entire GRID Architecture which receives all requests. There is only one hub in the selenium grid. Hub distributes the test cases across each node.

**5) What is a Node in Selenium Grid?**

– **Node** is a remote device that consists of a native OS and a remote WebDriver. It receives requests from the hub in the form of JSON test commands and executes them using WebDriver.

– There can be one or more nodes in a grid.

– Nodes can be launched on multiple machines with different platforms and browsers.

– The machines running the nodes need not be the same platform as that of the hub.

**6) What are the types of WebDriver API’s that are supported/available in Selenium?**

Selenium Webdriver supports most of the browser driver APIs like Chrome, Firefox, Internet Explorer, Safari and PhantomJS

**7) Which WebDriver implementation claims to be the fastest?**

HTML UnitDriver is the most light weight and fastest implementation headless browser for of WebDriver. It is based on HtmlUnit. It is known as Headless Browser Driver. It is same as Chrome, IE, or FireFox driver, but it does not have GUI so one cannot see the test execution on screen.

**8) What are the open source frameworks supported by Selenium WebDriver?**

Some of the popular open source frameworks supported by Webdriver are:

-TestNG

-JUnit

-Cucumber

-Robot Framework

-Appium

-Protractor

**9)** **What is the difference between Soft Assert and Hard Assert in Selenium?**

Hard Assert throws an *AssertException* immediately when an assert statement fails and test suite continues with next *@Test.*It marks method as fail if assert condition gets failed and the remaining statements inside the method will be aborted.

Soft Assert collects errors during *@Test*. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

**10) What are the verification points available in Selenium?**

Different types of verification points in Selenium are:

To check element is present

**if**(driver.findElements(By.Xpath(“value”)).size()!=0){

System.out.println(“Element is present”);

}**else**{

System.out.println(“Element is absent”);

}

To check element is visible

if(driver.findElement(By.Id(“submit”)).isDisplayed()){

System.out.println(“Element is visible”);

}**else**{

System.out.println(“Element is visible”);

}

To check element is enabled

if(driver.findElement(By.Id(“submit”)).isEnabled()){

System.out.println(“Element is enabled”);

}**else**{

System.out.println(“Element is disabled”);

}

To check text is present

if(driver.getPageSource().contains(“Text”)){

System.out.println(“Text is present”);

}**else**{

System.out.println(“Text is not present”);

}

**11) Why do we create a reference variable ‘driver’ of type WebDriver and what is the purpose of its creation?**

We create an instance of the WebDriver interface and cast it to different browser class using the reference variable ‘driver’. Then we can use different methods of the web driver interface like get(), getTitle(), close(), etc…to write automation code.

**12) What are the different types of exceptions you have faced in Selenium WebDriver?**

Different types of exceptions in Selenium are:

– NoSuchElementException

-NoSuchWindowException

-NoSuchFrameException

-NoAlertPresentException

-ElementNotVisibleException

-ElementNotSelectableException

-TimeoutException

**13) How to login into any site if it is showing an authentication pop-up for Username and Password?**

To work with Basic Authentication pop-up (which is a browser dialogue window), you just need to send the user name and password along with the application URL.

Syntax:

driver.get("http://admin:admin@yoururl.com");

**14) What is implicit wait in Selenium WebDriver?**

The implicit wait will tell the WebDriver to wait a certain amount of time before it throws a “No Such Element Exception.” The default setting of implicit wait is zero. Once you set the time, the web driver will wait for that particular amount of time before throwing an exception.

Syntax:

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

**15) What is Explicit Wait in Selenium WebDriver?**

Explicit waits are a concept from the dynamic wait, which waits dynamically for specific conditions. It can be implemented by the WebDriverWait class.

Syntax:

WebDriverWait wait = **new** WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id(“button”)));

**16) What is Fluent Wait in Selenium WebDriver?**

Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as NoSuchElementExceptions when searching for an element on the page.

Syntax:

Wait<WebDriver> wait1 = new FluentWait<>(driver)

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5))

.ignoring(NoSuchElementException.class);

WebElement element = wait1.until(new Function<WebDriver, WebElement>() {

@Override

**public** WebElement apply(WebDriver driver) {

**return** driver.findElement(By.id("firstName"));

}

});

**17) How to input text into the text box fields without calling the sendKeys()?**

We can use Javascript action to enter the value in text box.

Syntax:

JavascriptExecutor executor = (JavascriptExecutor)driver;

executor.executeScript("document.getElementById("textbox\_id").value='**new** value’”);

**18) How to clear the text inside the text box fields using Selenium WebDriver?**

Syntax:

driver.findElement(By.Id(“textbox\_id”)).clear();

**19) How to get an attribute value of an element using Selenium WebDriver?**

driver.findElement(By.Id(“button\_id”)).getAttribute(“text”);

**20) How to press Enter key on text box in Selenium WebDriver?**

driver.findElement(By.Id(“button\_id”)).sendKeys(keys.ENTER);

**21) How to pause a test execution for 5 seconds at a specific point?**

We can pause test execution for 5 seconds by using the wait command.

Syntax: driver.wait(5);

**22) Is Selenium Server needed to run Selenium WebDriver scripts?**

In case of Selenium WebDriver, it does not require to start Selenium Server for executing test scripts. Selenium WebDriver makes the calls between browser & automation script.

**23) What happens if we run this command driver.get(“www.google.com”);?**

It will load a new web page in the current browser window with the website url set to “www.google.com” . This is done using an http get operation, and the method will block until the load is complete.

**24) What is an alternative to driver.get() method to open a URL using Selenium WebDriver?**

We can use driver.navigate().To(“URL”) method to open a URL.

**25) What is the difference between driver.get(“URL”) and driver.navigate().to(“URL”) commands?**

driver.get() is used to navigate particular URL(website) and wait till page load.

driver.navigate() is used to navigate to particular URL and does not wait to page load. It maintains browser history or cookies to navigate back or forward.

**26) What are the different types of navigation commands in Selenium?**

Different navigation commands in selenium are:

– navigate().to();

-navigate().forward();

-navigate().back();

-navigate().refresh();

**27) How to fetch the current page URL in Selenium WebDriver?**

We can use the getCurrentUrl() method to get the current page URL.

driver.getCurrentUrl();

**28) How can we maximize browser window in Selenium WebDriver?**

We can use the maximize() method to maximize browser window.

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

**29) How to delete cookies in Selenium?**

We can use deleteAllCookies() method to delete cookies in selenium.

driver.manage().deleteAllCookies();

**30) What are the different ways for refreshing the page using Selenium WebDriver?**

Browser refresh operation can be performed using the following ways in Selenium:

– Refresh method

driver.manage().refresh();

– Get method

driver.get(“https://www.google.com”);

driver.get(driver.getCurrentUrl());

-Navigate method

driver.get(“https://www.google.com”);

driver.navigate.to(driver.getCurrentUrl());

-SendKeys method

driver. findElement(By.id("username")).sendKeys(Keys.F5);

**31) What is the difference between driver.getWindowHandle() and driver.getWinowHandles() in Selenium WebDriver and their return type?**

driver.getWindowHandle()  – To get the window handle of the current window. Returns a string of alphanumeric window handle

driver.getWinowHandles() – To get the window handle of all current windows. Return a set of window handles

**32) How to handle hidden elements in Selenium WebDriver?**

We can use the JavaScriptExecutor to handle hidden elements.

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("document.getElementById('displayed-text').value='text123'");

**33) How can you find broken links in a page using Selenium WebDriver?**

List<WebElement> elements = driver.findElements(By.tagName(“a”));

List finalList = **new** ArrayList();

**for** (WebElement element : elementList){

**if**(element.getAttribute("href") != **null**){

finalList.add(element);

}

}

**return** finalList;

**34) How to find more than one web element in the list?**

We can find more than one web element by using the findElements() method in Selenium.

List<WebElement> elements = driver.findElements(By.tagName(“a”));

**35) How to read a JavaScript variable in Selenium WebDriver?**

//Creating the JavascriptExecutor interface object by Type casting

JavascriptExecutor js = (JavascriptExecutor)driver;

//Perform Click on LOGIN button using JavascriptExecutor

js.executeScript("arguments[0].click();", button);

**36) What is JavascriptExecutor and in which case JavascriptExecutor will help in Selenium automation?**

JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium Webdriver.

In case, when selenium locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform a desired operation on a web element.

**37) How to handle Ajax calls in Selenium WebDriver?**

The best approach would be to wait for the required element in a dynamic period and then continue the test execution as soon as the element is found/visible. This can be achieved with WebDriverWait in combination with ExpectedCondition , the best way to wait for an element dynamically, checking for the condition every second and continuing to the next command in the script as soon as the condition is met.

WebDriverWait wait = **new** WebDriverWait(driver, waitTime);

wait.until(ExpectedConditions.visibilityOfElementLocated(locator));

**38) List some scenarios which we cannot automate using Selenium WebDriver?**

-Bitmap comparison is not possible using Selenium WebDriver.

-Automating Captcha is not possible using Selenium WebDriver.

-We can not read bar code using Selenium WebDriver.

-We can not automate OTP submission.

**39) How you build object repository in your project framework?**

We can build object repository using Page Object Model or Page Factory.

**40) What is Page Object Model (POM) and its advantages?**

Page Object Model is a design pattern for creating an object repository for web UI elements. Each web page in the application is required to have its own corresponding page class. The page class is thus responsible for finding the WebElements in that page and then perform operations on those web elements.

The advantages of using POM are:

-Allow us to separate operations and flows in the UI from verification – improves code readability

-Since the Object Repository is independent of test cases, multiple tests can use the same object repository

-Reusability of code

**41) What is Page Factory?**

Page Factory class in Selenium is an extension to the Page Object Design pattern. It is used to initialize the elements of the page object or instantiate the page objects itself.

Annotations in Page Factory are like this:

@FindBy(id = “userName”)

WebElement txt\_UserName;

OR

@FindBy(how = How.ID, using = “userName”)

WebElement txt\_UserName;

We need to initialize the page object like this:

PageFactory.initElements(driver, Login.class);

**42) What is the difference between Page Object Model and Page Factory?**

Page Object Model is a design pattern to create an Object Repository for web UI elements. However, Page Factory is a built-in class in Selenium for maintaining object repository.

**43) What are the advantages of Page Object Model?**

The advantages of using Page Object Model are:

-Allow us to separate operations and flows in the UI from verification – improves code readability

-Since the Object Repository is independent of test cases, multiple tests can use the same object repository

-Re-usability of code

**44) How can we use Recovery Scenario in Selenium WebDriver?**

We can develop Recovery scenarios using exception handling i.e. By using “Try Catch Block” within your Selenium WebDriver Java tests

**45) How to upload a file in Selenium WebDriver?**

Uploading files in WebDriver is done by simply using the sendKeys() method on the file-select input field to enter the path to the file to be uploaded.

driver.get(baseUrl);

WebElement uploadElement = driver.findElement(By.id("uploadfile\_0"));

// enter the file path onto the file-selection input field

uploadElement.sendKeys("C:\\newhtml.html");

**46) How to download a file in Selenium WebDriver?**

Step 1- Create a firefox Profile.

Step 2- set Preferences as per requirement.

Step 3- Open Firefox with firefox profile.

public **class** DownloadFiles {

public static **void** main(String[] args) {

// Create a profile FirefoxProfile profile=new FirefoxProfile();

// Set preferences for file type profile.setPreference("browser.helperApps.neverAsk.openFile", "application/octet-stream");

// Open browser with profile

WebDriver driver=**new** FirefoxDriver(profile);

// Set implicit wait driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

// Maximize window driver.manage().window().maximize();

// Open APP to download application driver.get("http://www.file.com/download\_file”);

// Click on download driver.findElement(By.xpath(“path”)).click();

}

}

**47) How to run Selenium WebDriver tests from command line?**

java -jar selenium-server.jar -htmlSuite "\*firefox" "http://10.8.100.106" "C:\mytestsuite\mytestsuite.html" "C:\mytestsuite\results.html"

**48) How to switch to frames in Selenium WebDriver?**

For switching between frames, use driver.switchTo().frame(). First locate the frame id and define it in a WebElement.

Ex:

WebElement fr = driver.findElementById("theIframe");

driver.switchTo().frame(fr);

**49) How to connect to a database in Selenium?**

Connection con = DriverManager.getConnection(dbUrl,username,password);

**50) How to resize browser window using Selenium WebDriver?**

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

**51) How to scroll web page up and down using Selenium WebDriver?**

To scroll using Selenium, you can use JavaScriptExecutor interface that helps to execute JavaScript methods through Selenium Webdriver.

JavascriptExecutor js = (JavascriptExecutor) driver;

//This will scroll the page till the element is found

js.executeScript("arguments[0].scrollIntoView();", Element);

**52) How to perform right click (Context Click) action in Selenium WebDriver?**

We can use Action class to provide various important methods to simulate user actions

//Instantiate Action Class

Actions actions = **new** Actions(driver);

//Retrieve WebElement to perform right click

WebElement btnElement = driver.findElement(By.id("rightClickBtn"));

//Right Click the button to display Context Menu

actions.contextClick(btnElement).perform();

**53) How to perform double click action in Selenium WebDriver?**

Action class method doubleClick(WebElement) is required to be used to perform this user action.

//Instantiate Action Class

Actions actions = **new** Actions(driver);

//Retrieve WebElement to perform double click WebElement

btnElement = driver.findElement(By.id("doubleClickBtn"));

//Double Click the button

actions.doubleClick(btnElement).perform();

**54) How to perform drag and drop action in Selenium WebDriver?**

//Actions class method to drag and drop

Actions builder = **new** Actions(driver);

WebElement from = driver.findElement(By.id("draggable"));

WebElement to = driver.findElement(By.id("droppable"));

//Perform drag and drop

builder.dragAndDrop(from, to).perform();

**55) How to highlight elements using Selenium WebDriver?**

// Create the JavascriptExecutor object

JavascriptExecutor js=(JavascriptExecutor)driver;

// find element using id attribute

WebElement username= driver.findElement(By.id("email"));

// call the executeScript method

js.executeScript("arguments[0].setAttribute('style,'border: solid 2px red'');", username);

**56) Have you used any cross browser testing tool to run Selenium Scripts on cloud?**

Below tools can be used to run selenium scripts on cloud:

-SauceLabs

-CrossBrowserTesting

**57) What are the DesiredCapabitlies in Selenium WebDriver and their use?**

The Desired Capabilities Class helps us to tell the webdriver, which environment we are going to use in our test script.

The setCapability method of the DesiredCapabilities Class, can be used in Selenium Grid. It is used to perform a parallel execution on different machine configurations. It is used to set the browser properties (Ex. Chrome, IE), Platform Name (Ex. Linux, Windows) that are used while executing the test cases.

**58) What is Continuous Integration?**

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

**59) How to achieve database testing in Selenium?**

//Make a connection to the database

Connection con = DriverManager.getConnection(dbUrl,username,password);

//load the JDBC Driver using the code

**Class**.forName("com.mysql.jdbc.Driver");

//send queries to the database

Statement stmt = con.createStatement();

//Once the statement object is created use the executeQuery method to execute the SQL queries

stmt.executeQuery(select \* from employee;);

//Results from the executed query are stored in the ResultSet Object. While loop to iterate through all data

**while**(rs.next()){

String myName = rs.getString(1);

}

//close the db connection

con.close();

**60) What is TestNG?**

TestNG is a testing framework inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easier to use. TestNG is an open source automated testing framework; where NG means NextGeneration.

**61) What are Annotations and what are the different annotations available in TestNG?**

Annotations in TestNG are lines of code that can control how the method below them will be executed. They are always preceded by the @ symbol.

Here is the list of annotations that TestNG supports −

**-@BeforeSuite:** The annotated method will be run only once before all tests in this suite have run.

**-@AfterSuite:**The annotated method will be run only once after all tests in this suite have run.

**-@BeforeClass:**The annotated method will be run only once before the first test method in the current class is invoked.

**-@AfterClass:**The annotated method will be run only once after all the test methods in the current class have run.

**-@BeforeTest:**The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.

**-@AfterTest:**The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.

**-@BeforeGroups:**The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

**-@AfterGroups:**The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

**-@BeforeMethod:**The annotated method will be run before each test method.

**-@AfterMethod:**The annotated method will be run after each test method.

**-@DataProvider:**Marks a method as supplying data for a test method. The annotated method must return an Object[ ][ ], where each Object[ ] can be assigned the parameter list of the test method. The @Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.

**-@Factory:**Marks a method as a factory that returns objects that will be used by TestNG as Test classes. The method must return Object[ ].

**-@Listeners:**Defines listeners on a test class.

**-@Parameters:**Describes how to pass parameters to a @Test method.

**-@Test:**Marks a class or a method as a part of the test

**62) What is TestNG Assert and list out some common assertions supported by TestNG?**

Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

Some of the common assertions are:

-assertEqual

-assertTrue

-assertFalse

**63) How to create and run TestNG.xml?**

Step 1: Add a new file to the project with name as testng.xml

Step 2: Add below given code in testng.xml

<suite name=“TestSuite”>

<test name=“Test1”>

<classes>

<**class** name=“TestClass” />

</classes>

</test>

</suite>

Step 3: Run the test by right click on the testng xml file and select Run As > TestNG Suite

**64) How to set test case priority in TestNG**?

We need to use the ‘priority‘ parameter, if we want the methods to be executed in specific order. TestNG will execute the @Test annotation with the lowest priority value up to the largest.

@Test(priority = 0)

public **void** One() {

System.out.println("This is the Test Case number One");

}

@Test(priority = 1)

public **void** Two() {

System.out.println("This is the Test Case number Two");

}

**65) What is parameterized testing in TestNG?**

To pass multiple data to the application at runtime, we need to parameterize our test scripts.

There are two ways by which we can achieve parameterization in TestNG:

* With the help of Parameters annotation and TestNG XML file.

@Parameters({“name”,”searchKey”})

* With the help of DataProvider annotation.

@DataProvider(name=“SearchProvider”)

**66) How to run a group of test cases using TestNG?**

Groups is one more annotation of TestNG which can be used in the execution of multiple tests.

public **class** Grouping{

@Test (groups = { “g1” })

public **void** test1() {

System.out.println(“**This** is group 1”);

}

@Test (groups = { “g2” })

public **void** test2() {

System.out.println(“**This** is group 2“);

}}

Create a testing xml file like this:

<suite name =“Suite”>

<test name = “Grouping”>

<groups>

<run>

<include name=“g1”>

</run>

</groups>

<classes>

<**class** name=“Grouping”>

</classes>

</test>

</suite>

**67) What is the use of @Listener annotation in TestNG?**

Listener is defined as interface that modifies the default TestNG’s behaviour. As the name suggests Listeners “listen” to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface. It allows customizing TestNG reports or logs. There are many types of TestNG listeners available:

-IAnnotationTransformer

-IAnnotationTransformer2

-IConfigurable

-IConfigurationListener

-IExecutionListener

-IHookable

-IInvokedMethodListener

-IInvokedMethodListener2

-IMethodInterceptor

-IReporter

-ISuiteListener

-ITestListener

**68) How can we create a data driven framework using TestNG?**

We can create data driven tests by using the DataProvider feature.

public **class** DataProviderTest {

private static WebDriver driver;

@DataProvider(name = "Authentication")

public static Object[][] credentials() {

**return** **new** Object[][] { { "testuser\_1", "Test@123" }, { "testuser\_2”, "Test@123" }};

}

// Here we are calling the Data Provider object with its Name

@Test(dataProvider = "Authentication")

public void test(String sUsername, String sPassword) {

driver = new FirefoxDriver(); driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://www.testqa.com");

driver.findElement(By.xpath(".//\*[@id='account']/a")).click();

driver.findElement(By.id("log")).sendKeys(sUsername);

driver.findElement(By.id("pwd")).sendKeys(sPassword);

driver.findElement(By.id("login")).click();

driver.findElement(By.xpath(".//\*[@id='account\_logout']/a")).click();

driver.quit();

}

}

**69) Where you have applied OOPS in Automation Framework?**

Abstraction – In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class. We utilize these locators in tests but we can’t see these locators in the tests. Literally we hide the locators from the tests.

Interface – WebDriver itself is an Interface. So based on the above statement WebDriver driver = new FirefoxDriver(); we are initializing Firefox browser using Selenium WebDriver. It means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object.

Inheritance – We create a Base Class in the Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc. We extend the Base Class in other classes such as Tests and Utility Class. Extending one class into other class is known as Inheritance.

Polymorphism – We use implicit wait in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.

Encapsulation – All the classes in a framework are an example of Encapsulation. In POM classes, we declare the data members using @FindBy and initialization of data members will be done using Constructor to utilize those in methods.

**70) How to handle Chrome Browser notifications in Selenium?**

// Create object of HashMap Class

Map<String, Object> prefs = **new** HashMap<String, Object>();

// Set the notification setting it will override the default setting

prefs.put("profile.default\_content\_setting\_values.notifications", 2);

// Create object of ChromeOption class

ChromeOptions options = **new** ChromeOptions();

// Set the experimental option

options.setExperimentalOption("prefs", prefs);

// pass the options object in Chrome driver

WebDriver driver = **new** ChromeDriver(options);

**71) Explain any Test Automation Framework?**

Testing frameworks are an essential part of any successful automated testing process. They can reduce maintenance costs and testing efforts and will provide a higher return on investment (ROI) for QA teams looking to optimize their agile processes. A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently. These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

**72) Tell some popular Test Automation Frameworks?**

Some of the popular test automation frameworks are:

-DataDriven

-KeywordDriven

-Hybrid

-Page Object Model

**73) Why Framework?**

Below are advantages of using an automation framework:

-Ease of scripting: With multiple Testers in a team, having an automation framework in place ensures consistent coding and that best practices are followed to a certain level. Standard scripting will result in team consistency during test library design and prevent individuals from following their own coding standards, thus avoiding duplicate coding.

-Scalable: Whether multiple web pages are being added or Objects or data, a good automation framework design is scalable when the need arises. A framework should be much easier to extend to larger projects.

-Modularity: Modularity allows testers to re-use common modules in different scripts to avoid unnecessary & redundant tasks.

-Easy to understand: Having an automation framework in place it is quick to transition (or understand) the overall architecture & bring people up-to-speed.

-Reusability: Common library files can be reused when required, no need to develop them every time.

-Cost & Maintenance: A well designed automation framework helps in maintaining the code in light of common changes like Test data, Page Objects, Reporting structure, etc.

-Maximum Coverage: A framework allows us to maintain a good range of Test data, i.e. coverage in turn.

-Better error handling: A good automation framework helps us catch different recovery scenarios and handle them properly.

-Minimal manual intervention: You need not input test data or run test scripts manually and then keep monitoring the execution.

-Easy Reporting: The reporting module within framework can handle all the report requirements.

-Segregation: A framework helps segregate the test script logic and the corresponding test data. The Test data can be stored into an external database like property files, xml files, excel files, text files, CSV files, ODBC repositories etc.

-Test configuration: Test suites covering different application modules can be configured easily using an automation framework.

-Continuous integration: An automation framework helps in integration of automation code with different CI tools.

-Debugging: Easy to debug errors

**74) Which Test Automation Framework you are using and why?**

Cucumber Selenium Framework has now a days become very popular test automation framework in the industry and many companies are using it because its easy to involve business stakeholders and easy to maintain.

**75) Mention the name of the Framework which you are using currently in your project, explain it in details along with its benefits?**

Framework consists of the following tools:

Selenium, Eclipse IDE,Junit, Maven, Cucumber

File Formats Used in the Framework:

-Properties file: We use properties file to store and retrieve different application and framework related configuration

-Excel files: Excel files are used to pass multiple sets of data to the application.

Following are the key components of the framework:

-PageObject : It consists of all different page classes with their objects and methods

-TestData: It stores the data files, Script reads test data from external data sources and executes test based on it

-Features: It consists of functional test cases in the form of cucumber feature files written in gherkin format

-StepDefinitions: It consists of different methods to implement each step of your feature files

-TestRunner: It is the starting point for Junit to start executing your tests

-Utilities: It consists of different reusable framework methods to perform different operations

-Reports: It consists of different test reports in different formats along with screenshots

-Pom xml: It consists of all different project dependencies and plugins

CUCUMBER

**1) What is Cucumber?**

Cucumber is a tool that supports Behaviour-Driven-Development(BDD) which is used to write acceptance tests for the web application. Cucumber can be used along with Selenium, Watir, and Capybara etc. Cucumber supports many other languages like Perl, PHP, Python, Net etc.

**2) What are the advantages of Cucumber?**

– Cucumber supports different languages like Java.net and Ruby.  
– It acts as a bridge between the business requirements and development code.  
– It allows the test script to be written without knowledge of any code, it allows the involvement of non-programmers as well.  
– It serves the purpose of end-to-end test framework unlike other tools.  
– Due to simple test script architecture, Cucumber provides code reusability.

**3) What are the 2 files required to execute a Cucumber test scenario?**

Step definition file and Test Runner file are the 2 files required to run a cucumber test scenario.

**3) What language is used by Cucumber?**

Cucumber uses Gherkin language. It is a domain specific language which helps you to describe business behavior without the need to go into detail of implementation. This text acts as documentation and skeleton of your automated tests.

**4) What is meant by a feature file ?**

The file, in which Cucumber tests are written, is known as feature files. The extension of the feature file needs to be “.feature”.

**5) What does a feature file consists of?**

Feature file consists of different test scenarios written using Gherkin language in Given/When/And/Then format.

**6) What are the various keywords that are used in Cucumber for writing a scenario?**

Most common keywords used in Cucumber are:  
– Feature  
– Background  
– Scenario  
– Given  
– When  
– Then  
– And  
– But

**7) What is Scenario Outline in Cucumber and its purpose?**

Same scenario can be executed for multiple sets of data using scenario outline. The data is provided by a tabular structure.

**8) What programming language is used by Cucumber?**

Cucumber supports a variety of different programming languages including Java, JavaScript, PHP, Net, Python, Perl, etc. with various implementations.

**9) What is the purpose of Step Definition file in Cucumber?**

A Step Definition is a Java method with an expression that links it to one or more Gherkin steps. When Cucumber executes a Gherkin step in a scenario, it will look for a matching step definition to execute.

To illustrate how this works, look at the following Gherkin Scenario:

Scenario: Login

Given user logins to the site

The user logins to the site part of the step (the text following the Given keyword) will match the following step definition:

public **class** StepDefinitions {

@Given("user logins to the site")

public **void** user\_login\_to\_site () {

System.out.println("User is logged into the site")

}

}

**10) What are the major advantages of Cucumber framework?**

– It is helpful to involve business stakeholders who can’t easily read code  
– Cucumber Testing focuses on end-user experience  
– Style of writing tests allow for easier reuse of code in the tests  
– Quick and easy set up and execution  
– Efficient tool for testing

**11) Provide an example of a feature file using the Cucumber framework.**

Feature: Login to site

Scenario: User login with valid credentials

Given user is on login page

When user enters valid username

And user enters valid password

**Then** login success message is displayed

**12) Provide an example of Scenario Outline using Cucumber framework?**

Feature − Scenario Outline

Scenario Outline − Login **for** facebook

Given user navigates to Facebook

When I enter Username as "<username>" and Password as "<password>"

**Then** login should be unsuccessful

Example:

| username | password |

| username1 | password1 |

| username2 | password2 |

**13) What is the purpose of Behaviour Driven Development (BDD) methodology in the real world?**

Behavior Driven Development is a software development approach that allows the tester/business analyst to create test cases in simple text language (English). The simple language used in the scenarios helps even non-technical team members to understand what is going on in the software project.

**14) What is the limit for the maximum number of scenarios that can be included in the feature file?**

You can have as many scenarios as you like, but it is recommended to keep the number at 3-5. Having too many steps in an example, will cause it to lose it’s expressive power as specification and documentation.

**15) What is the use of Background keyword in Cucumber?**

A Background allows you to add some context to the scenarios in the feature. It can contain one or more Given steps. It is run before each scenario, but after any Before hooks. In your feature file, put the Background before the first Scenario. You can only have one set of Background steps per feature. If you need different Background steps for different scenarios, you’ll need to split them into different feature files.

**16) What symbol is used for parameterization in Cucumber?**

The steps can use <> delimited parameters that reference headers in the examples table. Cucumber will replace these parameters with values from the table before it tries to match the step against a step definition.

**17) What is the purpose of Examples keyword in Cucumber?**

A Scenario Outline must contain an Examples (or Scenarios) section. Its steps are interpreted as a template which is never directly run. Instead, the Scenario Outline is run once for each row in the Examples section beneath it.

**18) What is the file extension for a feature file?**

Extension for a feature file is .feature

**19) Provide an example of step definition file in Cucumber.**

public **class** StepDefinitions {

@Given("user logins to the site")

public **void** user\_login\_to\_site () {

System.out.println("User is logged into the site")

}

}

**20) What is the purpose of Cucumber Options tag?**

@CucumberOptions annotation provides the same options as the cucumber jvm command line. For Example: we can specify the path to feature files, path to step definitions, if we want to run the execution in dry mode or not etc.

**21) How can Cucumber be integrated with Selenium WebDriver?**

We can integrate cucumber with selenium webdriver by adding all dependencies/jars related to selenium and cucumber in the project.

**22) When is Cucumber used in real time?**

Cucumber should be used for verifying the most important parts of the application using end-to-end tests. BDD should also be used to verify the wanted behaviour using integration tests. Importantly, the business should be able to understand these tests, so you can reduce uncertainty and build confidence in what you are building.

**23) Provide an example of Background keyword in Cucumber?**

Feature: Add items to shopping cart

Background: User is logged **in**

Given user navigate to login page

When user submits username and password

**Then** user should be logged **in** successfully

Scenario: Search a product and add it to shopping cart

Given user searches **for** dell laptop

When user adds the selected item to shopping cart

**Then** shopping cart should display the added item

**24) What is the use of Behaviour Driven Development in Agile methodology?**

The intent of BDD is to provide a single answer to what many Agile teams view as separate activities: the creation of unit tests and “technical” code on one hand, the creation of functional tests and “features” on the other hand.

**25) Explain the purpose of keywords that are used for writing a scenario in Cucumber?**

– Feature: The purpose of the Feature keyword is to provide a high-level description of a software feature, and to group related scenarios.  
– Scenario: In addition to being a specification and documentation, a scenario is also a test. As a whole, your scenarios are an executable specification of the system.  
– Given: steps are used to describe the initial context of the system – the scene of the scenario. It is typically something that happened in the past.  
– When: steps are used to describe an event, or an action. This can be a person interacting with the system, or it can be an event triggered by another system.  
– Then: steps are used to describe an expected outcome, or result.  
– Scenario Outline: This keyword can be used to run the same Scenario multiple times, with different combinations of values.  
– Background: It allows you to add some context to the scenarios in the feature. It can contain one or more Given steps.

PART-5

**1) What are the advantages of using TestNG?**

– It provides parallel execution of test methods  
– It allows to define dependency of one test method over other method  
– It allows to assign priority to test methods  
– It allows grouping of test methods into test groups  
– It has support for parameterizing test cases using @Parameters annotation  
– It allows data driven testing using @DataProvider annotation  
– It has different assertions that helps in checking the expected and actual results  
– Detailed (HTML) reports

**2) Can you arrange the below testng.xml tags from parent to child?**  
**<test>**  
**<suite>**  
**<class>**  
**<methods>**  
**<classes>**

<**suite**><**test**><**classes**><**class**><**methods**>

**3) What is the importance of testng.xml file?**

TestNG.xml file is an XML file which contains all the Test configuration and this XML file can be used to run and organize our test.

**4) What is TestNG Assert and list out common TestNG Assertions?**

Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. Some of the common assertions are:  
– assertEqual(String actual,String expected)  
– assertTrue(condition)  
– assertFalse(condition)

**5) What is Soft Assert in TestNG?**

SoftAssert don’t throw an exception when an assert fails. The test execution will continue with the next step after the assert statement.

**6) What is Hard Assert in TestNG?**

Hard Assert throws an AssertException immediately when an assert statement fails and test suite continues with next @Test.

**7) How to create Group of Groups in TestNG?**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">**

<**suite** name="TestNG Sample Test Suite">

<**test** name="TestNG Sample Test">

<**groups**>

<**define** name="include-groups">

<**include** name="sanity" />

<**include** name="integration" />

</**define**>

<**define** name="exclude-groups">

<**include** name="pass" />

<**include** name="fail" />

</**define**>

<**run**>

<**include** name="include-groups" />

<**exclude** name="exclude-groups" />

</**run**>

</**groups**>

<**classes**>

<**class** name="com.groups.TestNGGroupsExample" />

</**classes**>

</**test**>

</**suite**>

**8) How to exclude a particular test method from a test case execution using TestNG?**

<**suite** name="Sample Test Suite" verbose="1" >

<**test** name="Sample Tests" >

<**classes**>

<**class** name="com.test.sample">

<**methods**>

<**exclude** name="TestA" />

</**methods**>

</**class**>

</**classes**>

</**test**>

</**suite**>

**9) How to exclude a particular group from a test case execution using TestNG?**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">**

<**suite** name="Test Suite">

<**groups**>

<**run**>

<**exclude** name="integration"></**exclude**>

</**run**>

</**groups**>

<**test** name="Test">

<**classes**>

<**class** name="TestNGGroupsExample" />

</**classes**>

</**test**>

</**suite**>

**10) How to disable a test case in TestNG?**

@Test(enabled = **false**)

**public** **void** test() {

System.out.println("This test is disabled");

}

**11) How to ignore a test case in TestNG?**

@Test(enabled = **false**)

**public** **void** test() {

System.out.println("This test is ignored");

}

**12) What are the different ways to produce reports for TestNG results?**

There are two ways to generate a report with TestNG −

Listeners − For implementing a listener class, the class has to implement the org.testng.ITestListener interface. These classes are notified at runtime by TestNG when the test starts, finishes, fails, skips, or passes.

Reporters − For implementing a reporting class, the class has to implement an org.testng.IReporter interface. These classes are called when the whole suite run ends. The object containing the information of the whole test run is passed to this class when called.

**13) How to write regular expressions in testng.xml file to search @Test methods containing “smoke” keyword?**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">**

<**suite** name="testSuite">

<**test** name="test"> <**classes**>

<**class** name="sample">

<**methods**>

<**exclude** name="smoke.\*"/>

</**methods**>

</**class**>

</**classes**>

</**test**>

</**suite**>

**14) What is the time unit we specify in test suites and test cases?**

Time unit is Milliseconds

**15) List out various ways in which TestNG can be invoked?**

– Command Line  
– Maven  
– IDE

**16) How to run TestNG using command prompt?**

java -cp C:\Selenium\SampleTest\lib\\*;C:\Selenium\SampleTest\bin org.testng.TestNG testng.xml

**17) What is the use of @Test(invocationCount=x)?**

The invocationcount attribute tells how many times TestNG should run a test method. In this example, the method testCase will be invoked ten times:

@Test(invocationCount = 10)

**public** **void** testCase(){

System.out.println("Invocation method");

}

**18) What is the use of @Test(threadPoolSize=x)?**

The threadPoolSize attribute tells TestNG to create a thread pool to run the test method via multiple threads. With thread pool, it will greatly decrease the running time of the test method.

Example: Start a thread pool, which contains 3 threads, and run the test method 3 times

@Test(invocationCount = 3, threadPoolSize = 3)

**public** **void** testThreadPools() {

System.out.printf("Thread Id : %s%n", Thread.currentThread().getId());

}

**19) What does the test timeout mean in TestNG?**

While running test methods there can be cases where certain test methods get struck or may take longer time than to complete the execution than the expected. We need to handle these type of cases by specifying Timeout and proceed to execute further test cases / methods

@Test(timeOut=5000)

**public** **void** executeTimeOut() **throws** InterruptedException{

Thread.sleep(3000);

}

**20) What is JUnit?**

JUnit is an open source Unit Testing Framework for JAVA. It is useful for Java Developers to write and run repeatable tests.

**21) What are JUnit annotations?**

– @Test  
– @ParameterizedTest  
– @RepeatedTest  
– @TestFactory  
– @TestTemplate  
– @Disabled  
– @Tag  
– @AfterAll  
– @BeforeAll  
– @BeforeEach  
– @AfterEach  
– @TestInstance

**22) What is TestNG and what is its use?**

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use. It is used for:  
– Annotations  
– Run your tests in arbitrarily big thread pools with various policies available  
– Test that your code is multithread safe  
– Flexible test configuration  
– Support for data-driven testing  
– Support for parameters  
– Powerful execution model  
– Supported by a variety of tools and plug-ins  
– Embeds BeanShell for further flexibility  
– Default JDK functions for runtime and logging  
– Dependent methods for application server testing

**23) How is TestNG better than JUnit?**

– TestNG supports more annotations than Junit  
– TestNG supports ordering of tests but Junit doesn’t  
– TestNG supports various types of listeners using annotations but Junit doesn’t  
– TestNG reports are better than Junit

**24) How to set test case priority in TestNG?**

@Test (priority=1)

**public** **void** openBrowser() {

driver = new FirefoxDriver();

}

@Test (priority=2)

**public** **void** launchGoogle() {

driver.get("http://www.google.co.in");

}

**25) How to pass parameters through testng.xml to a test case?**

**public** **class** ParameterizedTest {

@Test

@Parameters("name")

**public** **void** parameterTest(**String** name) {

System.out.println("Parameterized value is : " + name);

}

}

<?xml version = "1.0" encoding = "UTF-8"?>

<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >

<suite name = "Suite1">

<test name = "test1">

<parameter name = "name" value="bijan"/>

<classes>

<**class** name = "ParameterizedTest" />

</classes>

</test>

</suite>

PART-6

**1) What is the difference between Manual and Automation Testing?**

– Manual Testing shows lower accuracy due to the higher possibilities of human errors. Automation Testing depicts a higher accuracy due to computer-based testing eliminating the chances of errors.  
– Manual Testing needs time when testing is needed at a large scale. Automation Testing easily performs testing at a large scale with the utmost efficiency.  
– Manual Testing takes more time to complete a cycle of testing, and thus the turnaround time is higher. Automation Testing completes a cycle of testing within record time and thus the turnaround time is much lower.  
– Manual Testing should be used to perform Exploratory Testing, Usability Testing and Ad-hoc Testing to exhibit the best results. Automation Testing should be used to perform Regression Testing, Load Testing, Performance Testing and Repeated Execution for best results.  
– Exploratory testing is possible in Manual Testing. Automation does not allow random testing.  
– The initial investment and ROI in the Manual testing is lower compared to Automation testing in the long run. The initial investment in the automated testing is higher and the ROI is better in the long run.

**2) What are the benefits of Automation Testing?**

– ROI: Even though the initial investment is high, return on investment over time is much better  
– Running tests: Tests can be run automatically or can be scheduled at a particular time  
– Fewer human resources: Manual testers are not required  
– Reusability: Scripts are reusable  
– Bugs: Automation helps you find bugs in the early stages of software development  
– Reliability: Automated testing is more reliable and way quicker  
– Parallel testing: Automated tests can be run parallelly on multiple devices

**3) Which Test cases needs to be automated?**

– Tests that need to be run against every build/release of the application, such as smoke test, sanity test and regression test.  
– Tests that need to run against multiple configurations — different OS & Browser combinations.  
– Tests that execute the same workflow but use different data for its inputs for each test run e.g. data-driven.  
– Tests that involve inputting large volumes of data, such as filling up very long forms.  
– Tests that take a long time to perform and may need to be run during breaks or overnight.

**4) What are the popular test automation tools for functional testing?**

– Selenium  
– Unified Functional Testing  
– Test Complete  
– Ranorex  
– Tosca

**5) What is the main purpose of Automation Testing?**

Automation testing is the best way to increase the effectiveness, efficiency and coverage of your software testing.

**6) What is the goal of Automation Testing?**

– To reduce Testing Cost and Time.  
– To reduce Redundancy.  
– To speed up the Testing Process.  
– To help improve Quality.  
– To improve Test coverage.  
– To reduce Manual Intervention.

**7) Why Selenium should be selected as a Test tool?**

– Language & Framework support: Selenium supports all major languages like Java, Python, JavaScript, C#, Ruby, and Perl programming languages for software test automation. Also, every Selenium supported language has dedicated frameworks which help in writing test script for Selenium test automation.  
– Open source: Being an open source tool, Selenium is a publicly accessible automation framework and is free, with no upfront costs.  
– Multi browser support: Selenium script is compatible with all browsers like Chrome, Safari, IE, etc …  
– Support across various OS: Selenium is yet a highly portable tool that supports and can work across different operating systems like Windows, Linux, Mac OS, UNIX, etc.  
– Ease of implementation: Selenium automation framework is very easy-to-use tool. Selenium provides a user-friendly interface that helps create and execute test scripts easily and effectively.  
– Reusability and Integrations: Selenium needs third-party frameworks and add-ons to broaden the scope of testing.For example, it can integrate with TestNG and JUnit for managing test cases and generating reports.  
– Parallel Test Execution: With the help of Selenium Grid, we can execute multiple tests in parallel, hence reducing the test execution time.

**8) What are the testing types that can be supported by Selenium?**

– Functional Testing  
– Regression Testing  
– Sanity Testing  
– Smoke Testing  
– Responsive Testing  
– Cross Browser Testing  
– UI testing (black box)  
– Integration Testing

**9) What are the limitations of Selenium?**

– Selenium does not support automation testing for desktop applications.  
– Since Selenium is open source software, you have to rely on community forums to get your technical issues resolved.  
– It does not have built-in Object Repository like UTF/QTP to maintain objects/elements in centralized location.  
– Selenium does not have any inbuilt reporting capability and you have to rely on plug-ins like JUnit and TestNG for test reports.

**10) What is the difference between Selenium IDE, Selenium RC and Selenium WebDriver?**

Selenium IDE:  
– It only works in Mozilla browser.  
– It supports Record and playback  
– Doesn’t required to start server before executing the test script.  
– It is a GUI Plug-in  
– Core engine is Javascript based  
– Very simple to use as it is record & playback.  
– It is not object oriented  
– It does not supports listeners  
– It does not support to test iphone/Android applications

Selenium RC:  
– It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.  
– It doesn’t supports Record and playback  
– Required to start server before executing the test script.  
– It is standalone java program which allow you to run Html test suites.  
– Core engine is Javascript based  
– It is easy and small API  
– API’s are less Object oriented  
– It does not supports listeners  
– It does not support to test iphone/Android applications

Selenium Webdriver:  
– It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.  
– It doesn’t supports Record and playback  
– Doesn’t required to start server before executing the test script.  
– It is a core API which has binding in a range of languages.  
– Interacts natively with browser application  
– As compared to RC, it is bit complex and large API.  
– API’s are entirely Object oriented  
– It supports the implementation of listeners  
– It support to test iphone/Android applications

**11) When should I use Selenium IDE?**

Because of its simplicity, Selenium IDE should only be used as a prototyping tool, not an overall solution for developing and maintaining complex test suites.

**12) What is Selenese?**

Selenese is the set of selenium commands which are used to test your web application. Tester can test the broken links, existence of some object on the UI, Ajax functionality, Alerts, window, list options and lot more using selenese.

**13) What is the difference between Assert and Verify commands?**

In case of the “Assert” command, as soon as the validation fails the execution of that particular test method is stopped and the test method is marked as failed. Whereas, in case of “Verify”, the test method continues execution even after the failure of an assertion statement.

**14) What is Same Origin Policy and how it can be handled? How to overcome same origin policy through web driver?**

Selenium uses java script to drives tests on a browser. Selenium injects its own js to the response which is returned from aut. But there is a java script security restriction (same origin policy) which lets you modify html of page using js only if js also originates from the same domain as html.

**15) How do you use findElement() and findElements()?**

WebElement login= driver.findElement(By.linkText("Login"));

List<WebElement> listOfElements = driver.findElements(By.xpath("//div"));

**16) Can Selenium handle window based pop up?**

No. Selenium cannot handle window based pop-up on its own but can use third party tools.

**17) How can we handle window based pop up using Selenium?**

We can handle window based popups using some third party tools such as AutoIT, Robot class.

**18) How can we handle web-based pop up using Selenium?**

Step 1: After opening the website, we need to get the main window handle by using driver.getWindowHandle();  
Step 2: We now need to get all the window handles by using driver.getWindowHandles();  
Step 3: We will compare all the window handles with the main Window handles and perform the operation the window which we need.

**19) How to assert title of the web page?**

String actualTitle = driver.getTitle();

String expectedTitle = "Title of Page";

assertEquals(expectedTitle,actualTitle);

**20) How to mouse hover on a web element using WebDriver?**

Actions builder = **new** Actions(driver);

builder.moveToElement(hoverElement).perform();

**21) How to retrieve CSS Properties of an element?**

driver.findElement(By.id("by-id")).getCssValue("font-size");

**22) What is JUnit?**

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks.

**23) What are JUnit annotations?**

Annotation is a special form of syntactic meta-data that can be added to Java source code for better code readability and structure.  
Some of them are:  
– @Before  
– @After  
– @BeforeClass  
– @AfterClass  
– @Ignore  
– @RunWith  
– @Test

**24) What is TestNG and what is its use?**

TestNG is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired from JUnit which uses the annotations (@). Using TestNG you can generate a proper report, and you can easily come to know how many test cases are passed, failed and skipped.

**25) How is TestNG better than JUnit?**

– In TestNG Annotations are easy to understand over JUnit.  
– TestNG enable you to grouping of test cases easily which is not possible in JUnit.  
– TestNG allows us to define the dependent test cases each test case is independent to other test case.  
– Parallel execution of Selenium test cases is possible in TestNG.

**Part-8**

**2) What are the top Selenium alternatives available for free?**

– Robot Framework  
– Ranorex  
– TestComplete  
– Tricentis Tosca  
– Soap UI

**3) What are different versions of Selenium available you have used and what are the additional features you have seen from the previous versions?**

Mostly worked with version 3 but now version 4 is going to be released. Following are additional features which are expected in new version:  
– Selenium 4 WebDriver is completely W3C Standardized  
– The Selenium IDE support for Chrome is available now  
– Improved Selenium Grid  
– Better Debugging  
– Better Documentation

**4) What is the principle difference between a Data-driven framework and a Keyword Driven Framework?**

Data driven framework includes different test data sources like flat files, databases or XML but Keyword Driven Framework involves business keywords which represent a feature or user actions.

**5) What are the two most common practices for automation testing?**

– Identify which test cases can be automated and which cannot be automated.  
– Do not rely completely on UI Automation.

**6) What is Test Driven Development (TDD) Framework?**

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the code is improved so that the tests pass.  
The following is an sequence of steps which are followed:  
– Add a test  
– Run all tests and see if it fails  
– Write the code  
– Run Tests  
– Refactor code  
– Repeat

**7) What is Behavior Driven Development (BDD) Framework?**

BDD is a software development process for teams to create simple scenarios on how an application should behave from the end user’s perspective. The goal of implementing BDD testing is to improve collaboration between stakeholders, such as developers, testers, product managers, and business analysts.

**8) What are the main traits of a good Software Test Automation framework?**

Maintainability, Reliability, Flexibility, Efficiency, Portability, Robustness, and Usability are main attributes of a good automation framework.

**9) What are the challenges have you faced with Selenium and how did you overcome them?**

– Handling dynamic content: We can use Implicit Wait or Explicit Wait to overcome this challenge  
– Handling popup up windows: We can use getWindowHandle and getWindowHandles methods to handle popups.  
– Handling alerts: We can use methods provided by Alert interface to handle an alert.  
– Handling false positives: We can use different Assertions to look out for false positives.  
– Synchronization issues: We can use different wait methods provided by selenium.  
– Window based dialogs: We can use AutoIt tool to automate window based dialogs

**10) What are the different components of Test Automation Framework?**

– Object Repository  
– Driver Script  
– Test Scripts  
– Function Library  
– Test data resources

**11) What are the benefits does WebDriver have over Selenium RC?**

– Webdriver architecture is simpler than RC  
– Webdriver is also faster than RC  
– WebDriver interacts with page elements in a more realistic way  
– WebDriver’s API is simpler than Selenium RC’s  
– WebDriver can support the headless HtmlUnit browser but RC cannot

**12) Which of the WebDriver APIs is the fastest and why?**

HTML Unit Driver is the fastest because it works on a simple HTTP request-response mechanism and doesn’t interact with the browser UI for execution.

**13) What is the command to bind a node to Selenium Grid?**

driver = new RemoteWebDriver(new URL(nodeURL),capability);

**14) Which of Java. C-Sharp or Ruby can we use with Selenium Grid?**

All the languages can be used with Selenium Grid.

**15) What are Selenium Grid Extras and the additional features does it add to Selenium Grid?**

Selenium Grid Extras is a project that helps you set up and manage your local Selenium Grid. Below are the additional features:  
– Killing any browser instance by name  
– Stopping any Process by PID  
– Moving mouse to specific location  
– Get Memory usage and disk statistics  
– Automatically upgrade WebDriver binaries  
– Restart node after a set number of test executions  
– Central storage of configurations for all nodes on the HUB server  
– Screenshots at the OS level

**16) Explain the concept of Object Repository?**

An Object Repository is a map between UI element and its locator. It can also be written as an Object Map between UI element and the way to find it.

**17) What is the difference between findElement() and findElements(), its return type and few examples of where you have used in Selenium Projects?**

findElement returns only first matching element but findElements returns a list of all matching elements.  
Return type for findElement is WebElement while for findElements is List<WebElements>  
findElements can be used in a scenario where we want to find all broken links in a webpage.

**18) Which method can be used to get the text of an element?**

We can use getText() method to get text of any element.

**19) How to check which check-box from multiple check-box options is selected previously using Selenium?**

isSelected() method is used to know whether the Checkbox is toggled on or off.

**20) What is the return type of isSelected() method in Selenium?**

Return type is boolean.

**21) What are the different methods which can be used to verify the existence of an element on a web page?**

– driver.findElements(By.xpath(“value”)).size() != 0  
– driver.findElement(By.id(id)).isDisplayed()  
– driver.findElement(By.id(id)).isEnabled()

**22) What is XPath Axes and what are the different Axes available?**

XPath axes search different nodes in XML document from current context node. XPath Axes are the methods used to find dynamic elements.  
– following: Selects all elements in the document of the current node  
– ancestor: The ancestor axis selects all ancestors element (grandparent, parent, etc.) of the current node  
– child: Selects all children elements of the current node  
– preceding: Select all nodes that come before the current node  
– following-sibling: Select the following siblings of the context node.  
– parent: Selects the parent of the current node  
– descendant: Selects the descendants of the current node

**23) How to fetch an element when its attributes are changing frequently?**

We can use different XPath methods like contains(), using or/and, starts-with, text(),ends-with

**24) What are the different ways to click on a button using Selenium?**  
– using click() method  
– using return key: sendKeys(Keys.Return)  
– using JavaScriptExecutor  
JavascriptExecutor js = (JavascriptExecutor) driver;  
js.executeScript(“document.getElementsByName(‘login’)[0].click()”);  
– using Actions class  
Actions actions = new Actions(driver);  
actions.moveToElement(button).click().perform();

**Part-9**

**1) How to handle Selenium WebDriver Exceptions?**

We can handle selenium exceptions by using try catch block methods of Java.

**try**{

driver.findElement(by.id("button")).click();

}

**catch**(NoSuchElementException e){

System.out.println("Element not present");

}

**2) There are four browser windows opened and you don’t have any idea where the required element is present. What will be your approach to find that element?**

– use getWindowHandles() method to get Window handles of all browser windows  
– use switchTo() method to switch to each browser window using the handle id  
– Find the element in each browser window and close the window if not present

**3) How do you handle an alert pop-up in Selenium?**

We can use the following methods to handle an alert in Selenium:

- dismiss()

driver.switchTo().alert().dismiss();

- accept()

driver.switchTo().alert().accept();

**4) How do you retrieve the text displayed on an Alert?**

**String** text = driver.switchTo().alert().getText();

**5) How do you type text into the text box on an Alert?**

driver.switchTo().alert().sendKeys("Text");

**6) Is Alert in Selenium an Interface or Class?**

Alert is an interface in Selenium.

**7) How do you handle frames in Selenium?**

We can switch to frames by following methods:  
– By Index  
driver.switchTo().frame(0);  
– By Name or Id  
driver.switchTo().frame(“id of the element”);  
– By Web Element  
driver.switchTo().frame(WebElement);

**8) Give an example for method overloading concept that you have used in Selenium?**

Implicit Wait in Selenium use method overloading as we can provide different Timestamp or TimeUnit like SECONDS, MINUTES, etc.

**9) How do you select a value from a drop-down field and what are the different methods available?**

We can select value from drop-down using methods of Select class. Following are the methods:  
– selectByVisibleText  
– selectByValue  
– selectByIndex

Select elements = new Select(driver.findElement(By.id("button"));

elements.selectByVisibleText("Selenium");

elements.selectByIndex(1);

**10) When your XPath is matching more than one element, how do you handle it to locate the required element?**

We can use index of the element to locate it or we can use different Xpath axes methods to locate the element like Following, Ancestor, Child, Preceding or Following-sibling

**11) How do you capture screen-shots in Selenium and what is the best place to have the screen-shot code?**

//Convert web driver object to TakeScreenshot

TakesScreenshot scrShot =((TakesScreenshot)webdriver);

//Call getScreenshotAs method to create image file

File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

//Move image file to new destination

File DestFile=new File(fileWithPath);

//Copy file at destination

FileUtils.copyFile(SrcFile, DestFile);

**12) Write the code for connecting to Excel files and other operations?**

XSSFWorkbook srcBook = new XSSFWorkbook("Demo.xlsx");

XSSFSheet sourceSheet = srcBook.getSheetAt(0);

**int** rownum=rowcounter;

XSSFRow sourceRow = sourceSheet.getRow(rownum);

XSSFCell cell1=sourceRow.getCell(0);

**13) How do you read and write into a PDF file?**

BufferedInputStream file = new BufferedInputStream("Path of PDF file");

PDFParser pdf = new PDFParser(file);

pdf.parse();

**String** text = new PDFTestStripper().getText(pdf.getPDDocument());

**14) What are the disadvantages of Selenium?**

– It supports only web applications and cannot automate desktop applications  
– No default reporting mechanism  
– No default object repository  
– Cannot automate captcha

**15) How do you debug your automation code when it is not working as expected?**

– Add breakpoints on the lines of code where it is not working

– Run code in debugging mode

– Use different actions like F7(Step Into), F8(Step Over), F9(Step Out) to debug the problem

**16) What are the end methods you use for verifying whether the end result is achieved by our Selenium automation scripts?**

We can use different assertion methods available in different test frameworks like TestNG or Junit.

**18) How do you implement collections in your framework?**

Collections can be used in framework in situations where you have to store large number of objects. For example, findElements() method returns a list of all matching elements.

**19) Give a scenario where inheritance is used in your framework?**

We create a Base Class in the Framework to initialize WebDriver interface, WebDriver waits, Property files, Excels, etc., in the Base Class. We extend the Base Class in other classes such as Tests and Utility Class.

**20) Give a scenario where interface is used in your framework?**

WebDriver is an interface and when we create an instance of the driver object to use its different methods.

**21) Write a code using JavascriptExecutor to scroll the web page?**

//This will scroll the web page till end.

js.executeScript("window.scrollTo(0, document.body.scrollHeight)");

**22) What is the use of property file in Selenium?**

Property file can be used to store the different web elements of an application or to store all the different application, framework configurations.

**23) How do you handle multiple browsers selection in Selenium?**

We can select different browsers in Selenium using TestNG framework.

**24) What do you use for reporting in your Selenium Project?**

We can use the default TestNG or Cucumber report. We can also use different reporting libraries like Extent reports.

**25) How Cross Browser testing is handled in Selenium?**

@BeforeTest

@Parameters("browser")

**public** **void** setup(**String** browser) **throws** Exception{

//Check if parameter passed from TestNG is 'firefox'

**if**(browser.equalsIgnoreCase("firefox")){

//create firefox instance

System.setProperty("webdriver.gecko.driver", ".\\geckodriver.exe");

driver = new FirefoxDriver();

}

//Check if parameter passed as 'chrome'

**else** **if**(browser.equalsIgnoreCase("chrome")){

//set path to chromedriver.exe

System.setProperty("webdriver.chrome.driver",".\\chromedriver.exe");

//create chrome instance

driver = new ChromeDriver();

}

testng.xml:

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">**

<**suite** name="TestSuite" thread-count="2" parallel="tests" >

<**test** name="ChromeTest">

<**parameter** name="browser" value="Chrome" />

<**classes**>

<**class** name="com.qascript.crossbrowsertests">

</**class**>

</**classes**>

</**test**>

<**test** name="FirefoxTest">

<**parameter** name="browser" value="Firefox" />

<**classes**>

<**class** name="com.qascript.crossbrowsertests">

</**class**>

</**classes**>

</**test**>

</**suite**>

**Part-10**

**1) Can Selenium automate Client Server applications?**

No Selenium cannot automate Client Server applications but there are other tools like Winium and AutoIt which can be used  
to automate desktop applications.

**2) What are the limitations of Selenium WebDriver?**

– It supports Web based applications only  
– Limited reporting capabilities  
– Handling captcha

**3) Tell me about the Selenium WebDriver architecture?**

WebDriver is made of following components:  
– Language Bindings  
– JSON Wire Protocol  
– Browser Drivers  
– Real Browsers

When a test script is executed with the help of WebDriver, the following tasks are performed in the background:

– An HTTP request is generated and it is delivered to the browser driver for every Selenium Command.  
– The HTTP request is received by the driver through an HTTP server.  
– All the steps/instructions to be executed on the browser is decided by an HTTP server.  
– The HTTP server then receives the execution status and in turn sends it back to the automation scripts.

Read more about this topic: [Click here](https://qascript.com/getting-started-with-selenium-webdriver/)

**4) How to identify the web elements?**

In order to identify WebElements accurately and precisely, Selenium makes use of following locators:  
-ID  
-Name  
-CSS  
-LinkText  
-XPath

**5) When do you go for an XPath?**

Although XPath can be used as a locator for any webelement, it is particularly useful when elements are dynamically changing or  
don’t have any unique properties.

**6) How to execute the tests on Firefox Browser in Selenium?**

System.setProperty("webdriver.gecko.driver",Path\_of\_Firefox\_Driver");

WebDriver driver = new FirefoxDriver(); //Creating an object of FirefoxDriver

driver.get("https://qascript.com)

**7) What is the difference between id and name?**

id is used to identify the HTML eleemnt through the DOM and is expected to unique within the page  
name correspons to the form element and identified what is posted back to server

**8) How to handle dynamic web elements in Selenium?**

Dynamic web elements can be handled in the following ways:  
– By starting text  
– containing text  
– By index  
– By following-sibling  
– By preceding text

**9) What is the default timeout of Selenium WebDriver?**

Default timeout is 30 seconds

**10) When do we use implicit and explicit waits in Selenium?**

– The implicit wait will tell to the web driver to wait for certain amount of time before it throws a “No Such Element Exception”.  
Once we set the time, web driver will wait for that time before throwing an exception.  
– The explicit wait is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or the maximum time exceeded  
before throwing an “ElementNotVisibleException” exception.

**11) How to select a date in a Calendar on a web page using Selenium?**

**public** **class** DatePicker

{

**public** **static** **void** main(**String**[] args) **throws** InterruptedException

{

**String** dot="9/October/2018";

**String** date,month,year;

**String** caldt,calmonth,calyear;

/\*

\* Split the String into String Array

\*/

**String** dateArray[]= dot.split("/");

date=dateArray[0];

month=dateArray[1];

year=dateArray[2];

ChromeDriver driver=new ChromeDriver();

driver.get("http://cleartrip.com");

driver.findElement(By.id("DepartDate")).click();

WebElement cal;

cal=driver.findElement(By.className("calendar"));

calyear=driver.findElement(By.className("ui-datepicker-year")).getText();

/\*\*

\* Select the year

\*/

**while** (!calyear.equals(year))

{

driver.findElement(By.className("nextMonth")).click();

calyear=driver.findElement(By.className("ui-datepicker-year")).getText();

System.out.println("Displayed Year::" + calyear);

}

calmonth=driver.findElement(By.className("ui-datepicker-month")).getText();

/\*\*

\* Select the Month

\*/

**while** (!calmonth.equalsIgnoreCase(month))

{

driver.findElement(By.className("nextMonth ")).click();

calmonth=driver.findElement(By.className("ui-datepicker-month")).getText();

}

cal=driver.findElement(By.className("calendar"));

/\*\*

\* Select the Date

\*/

List<WebElement> rows,cols;

rows=cal.findElements(By.tagName("tr"));

**for** (**int** i = 1; i < rows.size(); i++)

{

cols=rows.get(i).findElements(By.tagName("td"));

**for** (**int** j = 0; j < cols.size(); j++)

{

caldt=cols.get(j).getText();

**if** (caldt.equals(date))

{

cols.get(j).click();

**break**;

}

}

}

}

}

**12) Can I navigate back and forth in a browser using Selenium WebDriver?**

Yes. We can use Navigate method to move back and forth in a browser.

driver.navigate().forward();

driver.navigate().back();

**13) How to execute the Selenium scripts on different browsers?**

We can use a framework like TestNg or Junit and configure them to run Selenium Scripts on multiple browsers.

**14) What is the purpose of isDisplayed() function in Selenium WebDriver?**

The isDisplayed method in Selenium verifies if a certain element is present and displayed.  
If the element is displayed, then the value returned is true.If not, then the value returned is false.

**15) What is the difference between isDisplayed() and isEnabled() functions in Selenium WebDriver?**

isDisplayed() is capable to check for the presence of all kinds of web elements available.  
isEnabled() is the method used to verify if the web element is enabled or disabled within the webpage.

**16) Can you test flash images in Selenium?**

You can also automate the flash using Selenium web driver through the Flashwebdriver object and  
then call a method to operate flash object. You need to download flashwebdriver jar files

**17) What is a Framework?**

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results.

**18) How to select a third value from a drop-down field?**

Select select = new Select(listFrameworks);

select.selectByIndex(2);

**19) How to get columns from a table?**

WebElement table = driver.findElement(By.xpath("WebTableXPath"));

List<WebElement> totalRows = table.findElements(By.tagName("tr"));

**for**(**int** i=0;i<totalRows.size-1;i++){

List<WebElement> totalColumns = totalRows[i].findElements(By.tagName("td"));

}

**20) How many scripts are you writing and executing per a day?**

It is all dependent on the automation framework and the application under test.

**21) Which driver implementation will allow headless mode?**

HtmlUnit driver can be used to run tests in headless mode.

**22) Which reporting mechanism you have used in your Selenium projects?**

We used the Maven Cucumber Reporting plugin to generate detailed html reports.

**23) Why did you choose Selenium in your project, when there are so many tools?**

We chose Selenium because of the following reasons:  
– It is open source and free  
– It is easy to learn and setup  
– It is the most widely used and popular automation tool  
– All the web applications in our project are compatible with Selenium  
– Multi-browser and parallel testing is possible with Selenium

**24) How do you make use of JSON files in Selenium Grid?**

We can configure our hub and nodes using Json file in Selenium Grid.

**25) How to pause a test execution for 5 seconds at a specific point ?**

We can put a breakpoint on the line where we want to pause the test execution.

**Part-11**

**1) How to get the html source code of a particular web element using Selenium WebDriver?**

We can get the html source code of an element using getAttribute method.

driver.getAttribute("innerhtml");

**2) What are the different driver classes available in Selenium WebDriver API?**

Selenium WebDriver API consists of different types of Browser driver classes like ChromeDriver,IEDriver, FirefoxDriver, etc…

**3) What automation tools could be used for post-release validation with continuous intergration?**

We can use continuous monitoring tools such as Nagios and Splunk to perform post release validation.

**4) Does the latest version of Selenium WebDriver support Mobile Testing?**

Selenium directly doesn’t support Mobile testing but it is possible by the help of other tools like Appium.

**5) What is the major differences between XPath Expressions and CSS Selectors?**

Using XPath we can traverse both forward and backward whereas CSS selector only moves forward in HTML DOM.

**6) How to select a check box in Selenium?**

We can select a checkbox by clicking on it.

driver.findElement(By.id("chkbox")).click();

**7) How to verify whether the checkbox option or radio button is selected or not?**

By using isSelected() method.

driver.findElement(By.id("chkbox")).isSelected();

**8) What is the alternative way to click on login button?**

We can use JavaScriptExecutor to click on login button.

JavascriptExecutor js = (JavascriptExecutor)driver;

js.executeScript("arguments[0].click();", button);

**9) How can you find the value of different attributes like name, class, value of an element?**

By using the getAttribute() method.

**String** name = driver.findElement(By.id("login")).getAttribute("name");

**10) How to verify whether a button is enabled on the page?**

We can verify by using isEnabled() method.

driver.findElement(By.id("btn")).isEnabled();

**11) What kind of mouse actions can be performed using Selenium?**

Following mouse actions can be performed:  
– doubleClick(): Performs double click on the element  
– clickAndHold(): Performs long click on the mouse without releasing it  
– dragAndDrop(): Drags the element from one point and drops to another  
– moveToElement(): Shifts the mouse pointer to the center of the element  
– contextClick(): Performs right-click on the mouse

**12) What kind of keyboard operations can be performed in Selenium?**

Following keyboard actions can be performed:  
– sendKeys(): Sends a series of keys to the element  
– keyUp(): Performs key release  
– keyDown(): Performs keypress without release

**13) Can Bar Code Reader be automated using Selenium?**

It is not possible to automate bar code reader in Selenium.

**14) How to locate a link using its text in Selenium?**

We can use By.linkText() to locate a link using text in Selenium.

**15) Write the program to locate/fetch all the links on a specific web page?**

List<WebElements> allLinks = driver.findElements(By.tagName("a"));

**16) How can we run test cases in parallel using TestNG?**

By using the parallel attribute in testng.xml. The parallel attribute of suite tag can accept four values:

– tests: All the test cases inside <test> tag of Testing xml file will run parallel.  
– classes: All the test cases inside a Java class will run parallel  
– methods: All the methods with @Test annotation will execute parallel.  
– instances: Test cases in same instance will execute parall

**17) How do you get the height and width of a text box field using Selenium?**

We can get the height and width of a text box using getSize() method.

WebElement element = driver.findElement(By.id("txt"));

System.out.println(element.size());

**18) Which package can be imported while working with WebDriver?**

Following package can be imported: org.openqa.selenium.WebDriver

**19) What is the purpose of deselectAll() method?**

Clears all selected entries. This is only valid when the SELECT supports multiple selections.

**20) What is the purpose of getOptions() method?**

It Returns all the option elements displayed in the select tag for dropdown list.

**21) How to handle alerts in Selenium WebDriver?**

We can use switchTo().Alert() method to switch to the alert dialog and perform the below actions:

driver.switchTo.Alert().accept(); //To click on Ok button of alert

driver.switchTo().Alert().dismiss(); //To click on Cancel button of alert

**22) What is hybrid framework?**

Hybrid Framework is a combination two or more frameworks like Data-Driven, Modular or Keyword-Driven. It uses the best features of each framework to build a highly reusable and maintainable framework.

**23) Can you explain the line of code WebDriver driver = new FirefoxDriver();?**

It starts a new Firefox browser driver instance.

**24) What could be the cause for Selenium WebDriver test to fail?**

There could be many reasons for test failure. Some of them are listed below:  
– Driver is null or not found  
– Element is not found on the page  
– Element is present but not interactable  
– Page Synchronization issues

**25) What is the difference between @Factory and @DataProvider annotation?**

@DataProvider – A test method that uses @DataProvider will be executed multiple number of times based on the configuration provided in it.  
The test method will be executed using the same instance of the test class to which the test method belongs.  
@Factory – A factory will execute all the test methods present inside a test class using separate instances of the class.

**Part-12**

**1) Can we test APIs or web services using Selenium WebDriver?**

Selenium directly cannot test APIs or web services but we can use Java libraries like Rest Assured to do the same.

**2) How can we locate an element by only partially matching its attributes value in XPath?**

We can use contains() method in Xpath to partially match attribute value.

WebElement element = driver.findElement(By.xpath("//\*[contains(@id,'sub')]"));

**3) How can we locate elements using their text in XPath?**

We can use the text() method in XPath to locate elements.

WebElement element = driver.findElement(By.xpath("//\*[text(),'Selenium']));

**4) How can we move to parent of an element using XPath?**

We can use ancestor in Xpath to move to parent node of an element.

WebElement element = driver.findElement(By.xpath("//\*[@id='login']/ancestor::div[@class='button']));

**5) How can we move to nth child element using XPath?**

We can use child axis in Selenium to find all the children of the current node and  
then use index method to move to nth child element.

WebElement element = driver.findElements(By.xpath("//\*[@id='login']/child::\*")[2]);

**6) What is the syntax of finding elements by class using CSS Selectors?**

In Css Selector . represents a class identifier.

WebElement element = driver.findElement(By.CssSelector(".button"));

For a detailed view on all different locators in Selenium refer the following page – https://qascript.com/selenium-locators/

**7) What is the syntax of finding elements by id using CSS Selectors?**

In Css Selector # represents a class identifier.

WebElement element = driver.findElement(By.CssSelector("#login"));

For a detailed view on all different locators in Selenium refer the following page – https://qascript.com/selenium-locators/

**8) How can we select elements by their attribute value using CSS Selector?**

We need to provide the identifier type followed by the value in Css Selector to select an element.

**9) How can we move to nth child element using CSS Selector?**

driver.findElement(By.cssSelector("ul > li:nth-child(1)"));

**10) How can we submit a form in Selenium?**

We can submit a form using the submit() method in selenium.

driver.findElement(By.id("Login")).submit();

**11) How can we fetch a text written over an element?**

We can fetch text using the getText() method.

**String** text = driver.findElement(By.id("Login")).getText();

**12) What are some expected conditions that can be used in Explicit Waits?**

– elementToBeClickable()  
– elementToBeSelected()  
– presenceOfElementLocated()  
– visiblityOfElementLocated()

**13) How can we fetch the title of the page in Selenium?**

Using getTitle() method.

**String** title = driver.getTitle();

**14) How can we fetch the page source in Selenium?**

Using getPageSource() method.

**String** source = driver.getPageSource();

**16) How to check which option in the drop-down is selected?**

Using getFirstSelectedOption() method.

Select select = new Select(driver.findElement(By.xpath("//select")));

WebElement option = select.getFirstSelectedOption();

**String** defaultItem = option.getText();

System.out.println(defaultItem );

**17) How to handle HTTPS websites in Selenium? Does Selenium support them?**

You can handle HTTPS websites by handling SSL certificates in each browser using DesiredCapabilities.

**18) How to accept the SSL untrusted connection?**

For handling SSL error in Chrome, we need to use desired capabilities of Selenium Webdriver.  
The below code will help to accept all the SSL certificate in chrome, and the user will not  
receive any SSL certificate related error using this code.

// Create object of DesiredCapabilities class

DesiredCapabilities cap=DesiredCapabilities.chrome();

// Set ACCEPT\_SSL\_CERTS variable to true

cap.setCapability(CapabilityType.ACCEPT\_SSL\_CERTS, **true**);

// Set the driver path

System.setProperty("webdriver.chrome.driver","Chrome driver path");

// Open browser with capability

WebDriver driver=new ChromeDriver(cap);

**19) What is HtmlUnitDriver?**

HTML UnitDriver is the most light weight and fastest implementation headless browser for of WebDriver.  
It is based on HtmlUnit. It is known as Headless Browser Driver. It is same as Chrome, IE, or FireFox driver,  
but it does not have GUI so one cannot see the test execution on screen.

**20) What is the use of @Factory annotation in TestNG?**

A factory will execute all the test methods present inside a test class using separate instances of the class.  
It is used to create instances of test classes dynamically. This is useful if you want to run the test class any number of times.

**21) What are some common assertions provided by TestNG?**

– AssertTrue  
– AssertEqual  
– AssertFalse

**22) Name an API used for logging in Java?**

Log4J api is used for logging in Java.

**23) What is the use of logging in Automation?**

Logging helps in debugging errors and reporting purposes.

**24) Can Selenium Test an application on Android Browser?**

We have to use Appium to automate an application on Android browser.

**25) How to select a radio button in Selenium WebDriver?**

We can select a radio button using click() method.

driver.findElement(By.id("checkbox")).click();

# ****Part-24****

**1) What is Git? What is the difference between Git and GitHub?**

Git is a distributed version-control system for tracking changes in any set of files.

Git is a version control system to manage source code history but GitHub is a hosting service for Git repositories.

**2) What is Maven and explain about different Maven goals?**

Maven is a build automation tool used primarily for Java projects. Maven addresses two aspects of building software: how software is built and its dependencies.  
A plugin goal represents a specific task which contributes to the building and managing of a project. Below are some of the phases and default goals bound to them:  
compiler:compile the compile goal from the compiler plugin is bound to the compile phase  
compiler:testCompile it is bound to the test-compilephase  
surefire:test it is bound to testphase  
install:install it is bound to installphase  
jar:jar and war:war it is bound to package phase

**3) What is the difference between SVN & GIT?**

Git is a distributed version control system but SVN is a centralized version control system.

**4) How do you maintain source code in GIT?**

We maintain source code inside different repositories in GIT.

**5) What are the different plugins used for Maven? And it’s use?**

Some of the important plugins in Maven are:  
– clean: cleans up the target directories after the build is completed  
– compiler: compiles the Java Source Code  
– deploy: Deploys the built artifact to remote repository  
– failsafe: Runs the integration tests  
– install: Install the built artifact in the local repository  
– surefire: Runs the unit tests

**6) What’s the difference between plug-ins and dependencies?**

Plugins are core feature of Maven which allow reuse of common build logic across different projects.  
Dependencies are different archives which is required to build/compile/run current project.

**7) How do you define dependencies in your Maven Project?**

While working with a Maven project we need different libraries to run the project. These libraries can be automatically downloaded and managed by Maven dependencies.

**8) What is Maven and its advantages of using it in your Selenium Project ?**

Maven is a build management tool. Maven is used to define project structure, build and test management.

**9) What is the name of maven folder which contains all the libraries?**

External Libraries contain all the libraries.

**10) Explain the life cycle of Maven?**

Lifecycle of maven is the process of building and distributing a particular artifact. There are 3 built in build lifecycles – default, clean and site.

**11) Explain different Git commands?**

git add – Add file contents to the index  
git commit – Record changes to the repository  
git pull – Fetch from and integrate with another repository or a local branch  
git push – Update remote refs along with associated objects  
git init – Create an empty Git repository or reinitialize an existing one  
git clone – Clone a repository into a new directory

**12) In Maven, from where the jar files will get downloaded from?**

In maven jar files are downloaded from the maven central repository.

**13) In Maven, do we have to manually download and configure/update the required jar files?**

No. Maven automatically downloads and updates the required jar files and dependencies.

**14) What are the different phases of Maven?**

The default lifecycle comprises of the following phases :  
validate – validate the project is correct and all necessary information is available  
compile – compile the source code of the project  
test – test the compiled source code using a suitable unit testing framework.  
package – take the compiled code and package it in its distributable format, such as a JAR.  
verify – run any checks on results of integration tests to ensure quality criteria are met  
install – install the package into the local repository, for use as a dependency in other projects locally  
deploy – done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

**15) What is Jenkins?**

Jenkins is an open source Continuous Integration tool. It helps to automate the different stages of software development lifecycle like building, testing and deployment.

**16) How do you start Jenkins?**

We can start Jenkins by running the following command:  
java -jar jenkins.war –httpPort=8080

**17) How will you configure Jenkins?**

We need to configure the following in Jenkins:  
– Source Code Management  
– Build Triggers  
– Build Environment  
– Build  
– Post-build Actions

**18) What are the basic plugins you used in Jenkins?**

Some of the basic plugins in Jenkins are:  
– Email Extension  
– Git  
– JUnit  
– Maven Integration  
– NodeJS Plugin  
– SSH Plugin  
– Pipeline  
– Browserstack

**19) How to create a job in Jenkins?**

– Click on New Item in Jenkins dashboard  
– Enter an item name  
– Select type of project (Ex – Freestyle Project)  
– Enter configuration details for the job  
– Save the job

**20) On what platform JENKINS will work?**

Jenkins can run on Windows, Linux, MacOS and other Unix like operating systems.

**21) How do you manage reports in Jenkins?**

We can manage reports with the help of different reporting plugins like Junit, Cucumber, HTML.

**22) How to integrate your test with Jenkins?**

We can integrate our test by adding the git repository in the Source Code Management section and then invoking the maven goals in Build section to run the tests.

**23) How do you setup Maven Project in Jenkins?**

– Click on New Item in Jenkins dashboard  
– Enter an item name  
– Select type of project as Maven Project

**24) Jenkins Tool – Scheduled Batch Run?**

We can schedule our builds to run automatically in Jenkins by configuring the Build Triggers.  
Ex – Select Build Periodically and enter H/15 \* \* \* \* to run the build every 15 minutes.

**25) How to run the test scripts in Jenkins?**

We can run test scripts in Jenkins in the following ways:  
– By invoking Maven Goals in Build section  
– By executing shell/batch command