**1) What is Selenium Webdriver?**

Selenium WebDriver is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It directly communicates with the browser and controls it. Selenium WebDriver supports various programming languages like – Java, C#, PHP, Python, Perl, Ruby. and Javascript.

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

Selenium Grid is used to run tests on different machines against different browsers in parallel. 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 your 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

**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);

driver.manage().timeouts().implicitlyWait(Duration.ofSECOND(30));

**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) 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");

}

**62) 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

**63) 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

**64) 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

**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.abc.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) 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 it’s easy to involve business stakeholders and easy to maintain.

**74) 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

**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