**Q. Where you have applied OOPs concept in your Automation Framework? Which OOPs concept you have used in your framework?**

**Ans:-**

1. **Abstraction:** Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

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

1. **Interface:** An interface can have methods and variables just like the class but the methods declared in interface are by default abstract. We can achieve 100% abstraction and multiple inheritance in Java with Interface. Basic statement we have in selenium is WebDriver interface. Where we can create a reference instance of a chrome driver, Firefox Driver, Edge Driver.

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

1. **Inheritance:** The mechanism in Java by which one class acquires the properties (instance variables) and functionalities of another class is known as Inheritance. We extend the Base Class in other classes such as Tests and Utility Class.

Here we extend one class (Base Class like WebDriver Interface) into other class (like Tests, Utility Class) is known as Inheritance.

1. **Polymorphism:** Method having a different form.

* Method Overloading:We use **ImplicitlyWait** in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

**Action** class in TestNG is also an example of overloading.

**Assert** class in TestNG is also an example of overloading.

* Method Overriding:We use a method which was already implemented in another class by changing its parameters.

1. **Encapsulation:** The POJO class in java is an example of encapsulates.

**Q. Tell me some popular Test Automation Frameworks?**

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

There are different types of test automation frameworks and the most common ones are:

* **Modular Testing Framework:** In the modular testing framework, testers create test scripts module wise by breaking down the complete application under test into smaller, independent tests.
* **Data Driven Testing Framework:** The data-driven test automation framework is focused on separating the test scripts logic and the test data from each other. It allows us to create test automation scripts by passing different sets of test data. The test data set is kept in the external files or resources such as MS Excel Sheets, MS Access Tables, SQL Database, XML files, etc.,
* **Keyword Driven Testing Framework:** It is also known as table-driven testing or action word based testing. In Keyword-driven testing, we use a table format to define keywords or action words for each function or method that we would execute.
* **Hybrid Testing Framework:** Hybrid Test automation framework is the combination of two or more frameworks mentioned above. It attempts to leverage the strengths and benefits of other frameworks for the particular test environment it manages.
* **Behaviour Driven Development Framework:** The purpose of this Behavior Driven Development framework is to create a platform that allows everyone (such as Business Analysts, Developers, Testers, etc,) to participate actively. It requires increased collaboration between Development and Test Teams. Example- Cucumber, JBehave

**Q. Why do you prefer Selenium Automation Tool?**

Ans:-

1. Free and open source
2. Have large user base and helping communities
3. Cross-browser compatibility
4. Platform compatibility
5. Multiple programming languages support

# Q. Explain Test Automation Framework To The Interviewer?

# Ans:- Give summary based on below parameter whichever applicable to your framework.

# Common-utilities repo: Here we have added all common functionalities, Page Methods, Page Objects, and Utilities which will be used in all repos.

# Src/main/java:

# PageObjects pkg: It contains all Web Elements of pages

# PageMethods Pkg: It contains all pages based on the functionality

# Utilities pkg: It contains utilities which can handle the external data, API, file handling

# APICaller, APICallerHelper, BrowserActions, POJO class, Constants, JSONFilerParser, PropertyFileReader, CommonUtils

# Src/main/resources: It contains configURL.properties, which contains all environment gateway URL

# Properties folder

# Editprotocol Repo:

# Src/main/resources:

# Properties folder

# Src/test/java:

# StepDefinition pkg: It contains all the stepsDefinition of all features

# TestSuite pkg: It contains Runner class

# Src/test/resources:

# Driver folder

# Feature Folder

# TestData Folder

# Language: In our Selenium Project we are using Java language. Even though Selenium supports multiple languages, we are using Java language is just because most of the automation developers have knowledge on Selenium with Java.

# Type of Framework: In our project, we are using Cucumber BDD Framework.

# POM (Page Object Model): As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

# Packages: We have separate packages for Pages and Tests. All the web page related classes come under the Pages package and all the tests related classes come under Tests package. For example, Home Page and Login Page have separate classes to store element locators. For the login test, there would be a separate class which calls the methods from the Home Page class and Login Page class.

# Base Class: Base class deals with all the common functions used by all the pages. This class is responsible for loading the configurations from properties files, Initializing the WebDriver, Implicit Waits, Extent Reports, and also to create the object of FileInputStream which is responsible for pointing towards the file from which the data should be read.

# Utility Class: Utility class stores and handles the functions (The code which is repetitive in nature such as waits, actions, capturing screenshots, accessing excels, sending email, etc.,) which can be commonly used across the entire framework. The reason behind creating a utility class is to achieve reusability. This class extends the TestBase class to inherit the properties of TestBase in TestUtil.

# Properties file: This file (config.properties) stores the information that remains static throughout the framework such as browser-specific information, application URL, screenshots path, etc. All the details which change as per the environment and authorization such as URL, Login Credentials are kept in the config.properties file. Keeping these details in a separate file makes it easy to maintain.

# Screenshots: Screenshots will be captured and stored in a separate folder and also the screenshots of failed test cases will be added to the extent reports.

# Test Data: All the historical test data will be kept in an excel sheet (controller.xlsx). By using ‘controller.xlsx’, we pass test data and handle data-driven testing. We use Apache POI to handle excel sheets.

# TestNG: Using TestNG for Assertions, Grouping, and Parallel execution.

# Maven: Using Maven for build, execution, and dependency purpose. Integrating the TestNG dependency in the POM.xml file and running this POM.xml file using Jenkins.

# Version Control Tool: We use Git as a repository to store our test scripts.

# Jenkins: By using Jenkins CI (Continuous Integration) Tool, we execute test cases on a daily basis and also for nightly execution based on the schedule. Test Results will be sent to the peers using Jenkins.

# Extent Reports: For the reporting purpose, we are using Extent Reports. It generates beautiful HTML reports. We use the extent reports for maintaining logs and also to include the screenshots of failed test cases in the Extent Report.

# Q What are the utilities we should have in framework?

Ans:- We have added APICaller, APIHandler, Config.java (POJO), Constants interface, JSONParsor, Property file handler, Testdata generator

**Q. How to Capture Screenshot of Failed Test Cases Using Selenium WebDriver?**

Ans:- We can do this in 3 way:

1. **TestNG:** Add whole testcase steps in try block and screen capture statement in catch block

Example:-

@Test

**public** **void** swap2StringWithoutTempVar(){

**try** {

String str1="Test1";

String str2="Test2";

System.***out***.println("Before swapping: "+str1+" "+str2);

**int** length = str1.length();

str1=str1+str2;

System.***out***.println("After merging String: "+str1);

str2=str1.substring(0,length);

**int** length2= str1.length();

str1=str1.substring(length, length2);

System.***out***.println("After swapping: "+str1+" "+str2);

Assert.*assertEquals*(str2, "Test2");

}

**catch**(Exception e) {

File snap = ((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(snap,**new** File(".//test-output/failed.png"));

e.printStackTrace();

}

}

1. **TestNG:** Using ITestResult Interface. Create **AfterMethod** and add screen capture statement in it.

**Example:-**

@AfterMethod

**public** **void** takeScreenshot(ITestResult result) {

**if**(ITestResult.***FAILURE***==result.getStatus()) {

**try** {

File snap = ((TakesScreenshot)driver).getScreenshotAs(OutputType.***FILE***);

FileUtils.*copyFile*(snap,**new** File(".//test-output/failed.png"));

System.***out***.println("Successfully captured a screenshot");

}

**catch**(Exception e) {

e.printStackTrace();

}

}

driver.quit();

}

1. **Cucumber:** Using serenity framework. Add properties of screenshot in serenity.properties file

serenity.take.screentshots=AFTER\_EACH\_STEP

serenity.reports.show.step.details.screentshots=true

serenity.report.show.manual.tests=true

serenity.resized.image.width=1920

serenity.keep.unscaled.screenshots=false

**Q. How to take screenshot of failed testcase in ExtendReport?**

Ans:- https://www.softwaretestingmaterial.com/screenshots-extent-reports/

**Q. What are the different interfaces in selenium?**

**Ans:-** The following are the Interfaces and its abstract methods of Selenium webdriver.

1. **SearchContext->** findElement() and findElements()
2. **WebDriver->** get(String url), quit(), close(), getWindowHandle(), getWindowHandles(), getTitle(), getPageSource(), getCurrentUrl(), manage(), navigate(), switchto()
3. **TakesScreenshot ->** getScreenshotAs()
4. **JavascriptExecutor ->** executeScript() and executeAsyncScript()
5. **Navigation ->** to(), forward(), back(), refresh()
6. **OutputType ->** Fields: FILE, BYTES, BASE64, methods: convertFromBase64Png(), convertFromPngBytes()
7. **WebElement ->** clear(), click(), submit(), sendKeys(), getText(), getAttribute(), getCSSValue(), getLocation(), getSize(), isSelected(), isEnabled(), isDisplayed(), getTagName()
8. **Alert ->** accept(), dismiss(), getText(), sendKeys()
9. **Action ->** click(), clickAndHold(), doubleClick(), contextClick(), dragAndDrop(), moveToElement(), release(),
10. **ExpectedConditions -> elementToBeClickable(), elementToBeSelected(), presenceOfElementLocated(), visibilityOfElementLocated()**
11. **Timeouts ->** implicitlyWait(), setScriptTimout(), pageLoadTimeout()

**Q. What is JavaScriptExecutor? How JavaScriptExecutor works?**

Ans:- JavaScriptExecutor is an interface that is used to execute JavaScript through selenium webdriver. JavaScript is a programming language that interacts with HTML in a browser, and to use this function in Selenium, JavascriptExecutor is required. https://www.geeksforgeeks.org/javascriptexecutor-in-selenium/

**Q. Is JavaScriptExecutor class or interface?**

**Ans:-** JavaScriptExecutor is an interface. We can give reference of a driver to JavaScriptExecutor.

**Q. What type of test cases do you pick up to automate?**

**Ans:-** I focus on the test cases which should be executed in a repetitive manner such as regression test cases, smoke and sanity test cases, UI Testing, Functional test.

**Q. What type of test cases you won’t pick up to automate?**

**Ans:-** Before picking up the test cases to automate, I do check whether the application is stable or not. So based on this, I don’t pick up test cases when the AUT changes frequently and the test cases which I run rarely and run only one time. When I do usability and exploratory testing.

**Q. There is a button on webpage which is not clicking by selenium methods. What would be your approach to get that and click on that button?**

**Q. There is a button on webpage which is not clicking by selenium methods. What is the reason behind it why it is not clicking on button? And how will you resolve the issue?**

**Q. How do we click in Selenium when the click button is not working, and JavaScript is also not working?**

**Q. What are different types of framework? What is Data Driven framework? What is Keyword Driven framework? What is Hybrid driven framework?**

**Q. What is Page Object Model framework?**

**Q. What is POJO Class?**

**Q. What is Serialization and Deserialization in Rest Assured?**

**Q How to get response time in rest assured?**