1. Explain roles and responsibilities as an Automation Architect.
2. What code practices were followed during code review . Explain.- SOLID principles, KISS(Keep it simple stupid), DRY(Don’t repeat yourself): modularization- create functions and call functions as an when we needed.
3. Can you design automation framework from scratch(One which candidate has worked on his/her past experience) . Draw it in paint or other tools for better visualization and explain the components of the framework.
4. How did you implement OOPS concepts as a part of automation framework. Explain with an example- how and where **Inheritance**(BaseClass- Parentclass : driver initialization, properties initialization .)

Login extends BaseClass=> POM for your login functionality

LoginTest extends BaseClass)

1. , **Abstraction**(hiding internal implementation details to the world- Interfaces), **Polymorphism**(Method overloading –one method whose name is same but it can have different parameters,data types, number of arguments, sequence of arguments.)- Constructors overloading

SimpleFramework()

SimpleFramework(int name, int desc)

Method Overriding- Overriding definition of your parent class(SuperClass) – getDetails() method in your subclass(ChildClass) getDetails()

ParentClass:

getDetails() {

Syso(“In Parent class”);

}

Uparrow ChildClass:

getDetails() {

Syso(“In Child class”);

.......

}

1. and method overriding), **Encapsulation**(private variables- getter and setter), **Interfaces**(WebDriver- ChromeWebDriver,IEWebDriver), **Class and Objects** were used.
2. Solve 1-2 coding questions in Java by interviewer. It usually is around Medium complexity mostly asked from LeetCode or GeeksforGeeks interview questions list.
3. **Design patterns, code practices, debugging skills**.

Design patterns used in automation:

1. Page Object Model Design Pattern, Page Factory(Object repository- create variable for locators and create methods).
2. Factory Design Pattern
3. Facade Design Pattern
4. Singleton Design Pattern
5. Fluent Page Object Model
6. Data-Driven Testing – TestNG- Data Provider
7. Fluent Interface Pattern
8. Structural design pattern.

**Best Coding practices:**

1. **No hardcoding. (any locator, url, username, password, testdata, parameters- passed)- these should come from source: excel/xml/properties file/database...**
2. **Use of comments and functional documentation comment.**
3. **Not making use of inheritance when not required.- code reusability: Composition, Aggregation, Inheritance, Interfaces and have classes implements**

**Department**

**Students**

1. **Making good use of interfaces.**
2. **Avoid tight code coupling. Code is dependent on another code--- one fails , other is definitely going to fail. Do loose coupling.**
3. **Keep code clean and simple, avoid clustering of code. Not leaving any line between and continuosly lot of lines code in a method.**
4. **Follow KISS(Keep it simple stupid) , SOLID, DRY(Dont repeat yourself) principles,**

**SOLID:**

**S: substitution principle.**

**O- Open /Closed Princicple**

**L- Liskow’s substitution pricincple**

**I: Interface segregation**

**D- Dependency inversion.**

1. **Proper indentation.**
2. **Single purpose rather than making one class/method do all the task.**
3. **Avoid logical issues like null pointer exception with thorough unit testing.**
4. **Use of proper naming conventions.**
5. **Keep project structure relevant , simple and to-purpose.**
6. Write xpath to find a cell that contains the word “Electronics” from a given table.
7. Explain different types of exceptions encountered while doing automation.
8. Write a code to launch a browser, open a website and perform signup on that website and perform log out using Cucumber BDD approach.