**Java**

1. How would you handle Exception?

* I would use try-catch-finally approach to handle the Exception
* I would put my code that might generate an exception inside a try-catch block.
* I can also use the throws keyword. BUT it does mean that anyone that calls my method now needs to handle it too!

1. Difference between RuntimeException and CheckedException in Java?

* Exception are divided in two categories Runtime (unchecked) Exception and CheckedException.
* Main difference between RuntimeException and CheckedException is that, it is mandatory to provide try-catch to handle CheckedException while in case of RuntimeException is not mandatory.
* Some of the most common Exception like NullPointerException, ArrayIndexOutOfBound, ClassNotFoundException, IOException.

1. How do you use an abstract class in your project give me an example?

These concepts are commonly used in framework development. Abstract class is used in defining a common super class while writing Page Object Model layer of the framework. We usually create an abstract class named BasePage to have all common members for every page written in this class example getPageTitle().

Then each Page class (HomePage, LoginPage, DashboardPage etc.) inherit from BasePage. Sometimes one may need to change the behavior of methods implemented in superclass. So, subclass has freedom to override that method where we use polymorphism. This is how we use Abstract class in real projects.

**Selenium**

1. What is the difference between isDisplayed(), isEnabled(), and is Selected() method in Selenium WebDriver?

* isDispalyed() => verifies the presence of a web element within the web page. If found => true, If not found => false
* isDisplayed() => checks for the presence of all kinds of web elements available
* isEnabled() => verifies if the web element is enabled or disabled within the web page.
* isEnabled() => is primarily used with buttons
* isSelected() => verifies if the web element is selected or not
* isSelected() => is used with radio buttons, dropdowns and checkboxes.

----------------

1. How to do Drag And Drop ?

Actions action = new Actions(driver);

action.clickAndHold(driver.findElement(By.id("item")))

.moveToElement(driver.findElement(By.id("destination")))

.release().build()

.perform();

1. How to use Excel?

FileInputStream ExcelFile = new FileInputStream(path);

excelWBook = new XSSFWorkbook(ExcelFile);

excelWSheet = excelWBook.getSheet(sheetName);

cell = excelWSheet.getRow(rowNum).getCell(colNum);

# API

1. HTML Status Codes?

● 1xx → Informational

● 2xx → Success (request was accepted successfully) (200→ Ok, 201→ Created, 202→ Accepted, 204→ No Content)

● 3xx → Redirection

● 4xx → Client Error (400-Bad Request, 401-Unauthorized, 403-Forbidden, 404-Not Found, 405-Method not Allowed)

● 5xx → Server Error (500-Internal server Error, 502-Bad Gateway, 501-Not implemented, 503-Service Unavailable)

1. What first thing you check when you get a response?

● Status quote (200 always mean Ok)

● We always check the 404 means not found

● rest-assured.io==> for automation to find the ECS machine in search type remote Desktop

1. Http methods and request types

● Get does not requires body

● Put requires body means UPDATE information

● Post requires body means CREATE information

● Delete does not requires body

● GET -> READ , POST -> CREATE, PUT -> UPDATE, DELETE -> DELETE

● POST VS PUT

1. Serialization and Deserialization

● Serialization; when we MAP a Java object to API JSON format (CONVERT JAVA OBJECT TO JSON);

○ Java object (POJO(Plain Old Java Object), BEANS) → MAP it to API JSON/XML

○ When we have an object from a class and MAP it to a JSON format in our RESTful API

{make: “Toyota”,

Model: “Camry” }

Car car = new Car();

car.setMake(“Toyota”);

car.setModel(“Camry”);

given().body(car).when().post(uri)

● Deserialization; API JSON/XML → MAP it to Java Object (JSON TO JAVA OBJECT)

Car car2 = new Car();

car2=when().get(uri).body.as(car.class);

car.setMake(“Toyota”);

car.setModel(“Camry”);

1. What is JSON?

● It is JavaScript Object Notation (is a minimal, readable format for structuring data.)

● It is used primarily to transmit data between a server and web application, as an alternative to XML.

● Basically, a lightweight version of XML

● In Key: Value format

● Key is always in double quotes and value if string its double quotes and if numbers no quotes

● It is purely based on http protocol, - so it hits the link on the browser and see the results

1. Do you know swagger? What is swagger?

● Swagger is an open-source software framework backed by a large ecosystem of tools that helps developers design,

build, document, and consume RESTful Web services.

● Swagger allows you to describe the structure of your APIs so that machines can read them.

● The ability of APIs to describe their own structure is the root of all awesomeness in Swagger

● similar to xml schema but for Json

1. json vs gson

● JSON is a format which has key and values

● GSON is a process of converting

○ from java to json(serialization),

○ from json to java(deserialization)

**Git & GitHub**

1. What is GitHub?

● Version control system

● Keeps track of new/old version of documents

● Manages/stores set of files

2. What is repository?

● Folder where the files are saved and

● It may contain single, collections of files, or single projects.

3. What is Remote & Local Repository?

● Remote Repository: Host on server(GITHUB) Our changes go from local to remote repo

● Local Repository: Typically, on your computer -Our changes are done here consist of Working Directory, index and HEAD

4. What are Git commands?

● Add: add to staging area

● Commit: add from working directory and local repo

● Push: add to remote repo

● Pull: take changes from remote to working directory

● Clone with url: clones url into directory

● Git version: give you version of git

● Git status: shows you what branch you’re on, any changed files that aren’t tracked

○ Origin: name of remote

○ Master: name of branch

● Git add:

○ Adding to staging area

○ Recursive add

○ Adds everything

● git commit -m: “message will apply for all files”

● git push: origin nameOfBranch

● git ignore:

○ Notepad.gitignore => In the notepad add files you don't want to add to staging area

○ You must push the .gitignore file to repo in order for the files you want to be ignored.

○ Some files don't matter and shouldn’t be pushed to git

● checkout branch => Git checkout -b nameOfBranch master

**SQL**

Please explain what will be the output for the query below

# 

# 

# 

**Tell me About Your Framework**

In my current project I am using Cucumber BDD (Behavior Driven Development) framework to automate our test scripts. Cucumber is a tool to use to make automated testing understandable to non-technical team members.

I pull data from the excel document by adding Apache POI libraries to pom.xml file. In my project, I am using Java Language.

I use Selenium Webdriver to automate the browser. I also use Maven for build, execution, dependency and plugins purposes. I use Junit as a testing tool.

My framework was created based on POM(Page object model) by creating separate packages to organize our java code. In this way, I am able to keep our code cleaner and more understandable.

Basically, I have 6 major packages in the framework.

Pages, feature files, Step definitions, runner class, hook class, and utilities packages. Let me explain these components.

1- Pages: In this package, I created a separate Java class for each page of my application where I store all the elements of that page as well as related methods. I use @Findby annotations to locate the elements. In my framework, there are 25 pages total and in each page, there is a constructor that includes pagefactory.initelements to initialize the driver.

2- Feature Files: For each scenario(positive or negative) I created Cucumber feature files

where I used GHERKIN language in order to describe my test scenarios. Gherkin language is that Gherkin keywords + simple English. It has some keywords such as: Feature, Scenario, Scenario Outline, Examples, Given, And, When, Then… Tags is to run the specific scenario.

3- Step Definitions: Step Definition is a small piece of code or a java method in a class with an annotation above it. An annotation followed by the pattern is used to link the Step Definition to all the matching Steps, and the code is what Cucumber will execute when it sees a Gherkin Step.

4- Runner class:

@RunWith(Cucumber.class)

@CucumberOptions(

format = {"pretty","json:target/cucumber.json","html:target/cucumber-html"}

features = {"src/test/resources/features"},

glue = {"com/myProject/stepDefinitions"},

tags= {"@Test111"},

dryRun = false, ) public class Run { }

@CucumberOptions are used to set some specific properties for the Cucumber test and it comes from

Cucumber.

Features Options helps Cucumber to locate the Feature file in the project folder structure Glue options is almost the same thing as Features Option, but the only difference is that it helps Cucumber to locate the Step Definition file.

Format Option is used to specify different formatting options for the output reports.

dryRun option can either be set as true or false. If it is set as true, it means that Cucumber will only

checks that every Step mentioned in the Feature File have corresponding code written in Step

Definition file or not. So, in case any of the function is missed in the Step Definition for any Step in Feature File, it will give us the message. Default value as false.

@RunWith

Now that we have defined the test it's time to run our test. But before we do that we have to add a class

for running our tests. Cucumber uses the Junit framework to run.

First import statement ‘org.junit.runner.RunWith‘ imports @RunWith annotation from the Junit

class. @RunWith annotation tells JUnit that tests should run using Cucumber class present in

‘Cucumber.api.junit‘ package.

Now give it a run by Right Click on TestRunner class and Click Run As > JUnit Test. Cucumber will run the script by using Junit and the result will be shown in the left-hand side project explorer window in JUnit tab.

5- Hook Class:

Cucumber supports hooks, which are blocks of code that run before or after each scenario. You can

define them anywhere in your project or step definition layers, using the

methods @Before and @After. Cucumber Hooks allows us to better manage the code workflow and

helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to

perform our scenarios or tests.

Cucumber supports only two hooks (Before & After) which works at the start and the end of the test

scenario. As the name suggests, @before hook gets executed well before any other test scenario,

and @after hook gets executed after executing the scenario.

6- Utilities Package: In the utilities folder, I store my reusable codes in the utilities package such as browser utils, DatabaseUtils, ExcelUtil, Driver, ApplicationConstants, Pages, configuration Reader.

Browser Utils: SwitchToWindow, HoverOver, getElementsText, waitForVisibility, SelectFromDropdown, etc.

DatabaseUtil: Create database connection and CRUD(Create, read, update, delete) actions with Database.

ExcelUtils: Methods to help pull data from excel documents.

Driver: Driver object instance is stored here.

ApplicationConstants: Contains expected string variables.

Pages: Instance is created for each page by using a singleton method.

ConfigurationReader: Credentials, browser types, and urls are stored here.

# 

# Real Interview Questions

1. Are you getting paid in your current position ?
2. Have you worked in a fast paced environment before ? (They have 2 weeks sprint)
3. Could you tell me about your previous company and project ?
4. Are you in 1099 ? What is your hourly rate ?
5. Are you interested in a permanent W-2 with benefits ?
6. Why did you leave your previous project (not current) ?
7. How soon can you start ?
8. Have you ever done performance and load testing before ?
9. What is your strength and what is your weakness ?
10. How to Run cucumber test on parallel ?
11. How do you take screenshot in your current framework ?
12. What is Test Base Class ?
13. What is CICD ?
14. How do you schedule your smoke tests in Jenkins ?
15. How do you run your smoke tests ?
16. What is the Page object model design ?
17. What is Final variable ?
18. What is Final Class ?
19. What is Inheritance ?
20. What is Overloading ?
21. What is Finally ?
22. How do you rate yourself in terms of Java ?
23. What is the difference between post and put ?
24. How did you get in IT?
25. What is the TestNg?Annotations?