Framework IQ:

1. **Automation challenges and how did you solve them? What are the challenges you face when running automation scripts? Overall challenges?**

Scenario 1: Maintenance of Web Elements

When I just joined the company we faced issue with maintaining elements. Framework was not properly structured and maintenance of it was taking more time than automation of new features. With the help of my team I implemented POM concept in our framework. I gave a demo of the new design to the entire team and once it was approved we continued our automation. As of now we have complete robust framework that is easily maintainable and understandable by any automation engineer that joins the team.

Scenario 2: Cross browser Testing issues

In every project it is required to perform multi-browser testing to make sure that the functionality is working as expected with every browser to give equal user experience to all of the wide range of Users. Our selenium test scripts were executed properly in Chrome Browser, but in IE browser we faced so many issues like timeout problems/Synchronization problems...We found that IE Browser is slower than Chrome browser in Test case execution so we implemented both implicit and explicit wait for our scripts and also changed some of the locator from xpath into the css. As we know Css path is faster than xpath so we just reduced the random failure and slowness of execution. On top of that since we work in agile and we work closely with developers whenever they were working on new features I asked them to provide ids for the elements.

Scenario 3: Dynamically changing elements on the web page

To resolve that issue we can go for XPath, where I can use contains or start with:

```
//div[contains(@id,'post-body')]
//div[starts-with(@id,'post-body')]
```

Also I can create my own custom XPath by using parents or preceding/following siblings:

```
//label[text()='DateofBirth']/following-sibling::img[@class='ui-datepicker-trigger']
```

Scenario 4: Stale Element Reference Exception:

In my current application at sometimes we were getting errors saying Stale Element Reference Exception for some specific elements. After analyzing failure issues I implemented WebDriver waits and Javascriptexecutor to handle those errors.

Scenario 5: Working with dynamic values in WebTable

Data in Web Table mostly is always dynamic, so we cannot hardcode any cells or rows elements. Everytime we work with WebTable we will have to loop through each row and cell

```
//Grab the table
WebElement table = driver.findElement(By.id("divListView"));
// Now get all the TR elements from the table
List<WebElement> allRows = table.findElements(By.tagName("tr"));
// And iterate over them, getting the cells
for (WebElement row : allRows) {
        List<WebElement> cells = row.findElements(By.tagName("td"));
        // Print the contents of each cell
        for (WebElement cell : cells) {
                System.out.println(cell.getText());
                if (cell.getText().contentEquals("required cell we want to click")){
                        cell.click();
                }
        }
}
```

Scenario 6: Working with frames

In some application for better visibility developer use frame concept in web pages. In this case, if your element exists in frames then we have to switch to frame first then we have to perform our operation.

Until you are inside the frame you can not perform any operation so once we are done with frame then switch To parent window:

driver.switchTo().defaultContent();

This is overall for Selenium

Scenario 7: Any reporting related capabilities, you need to depend on third party tools

Selenium testing does not provide reporting functionality so we need to depend on the third party tools Like TestNG for generating reports if we work with TestNG framework If we work with cucumber framework we are using cucumber reports for reports generation which we use maven surefire plugin for this.

Scenario 6: Selenium work only for Web Based application

2. **What you have done to improve the performance of selenium framework?**

- Create reusable component
- Write optimise function with precise logic
- Follow coding standard
- Write locator to handle dynamic elements
- Provide data in external files such as excel, txt, property files

3. **How many changes do you need to make to manage your Selenium scripts? How do you maintain your test scripts and how frequently you have to modify them?**

Is is all depend on types of changes. As an example if any webelement change since we are maintaining POM design we only need to update that 1 web element in that particular class page. However, if existing functionality change then whatever functions we have created that we need to update as per new changes.

Whenever we got new changes and test fails that time we refine & modify test scripts.

4. **What is the page object model in Selenium? What are the advantages of Page Object Model?**

Page Object Model is a design pattern which has become popular in test automation for enhancing test maintenance and reducing code duplication. A page object is an object-oriented class that serves as an interface to a page of your AUT.

The tests then use the methods of this page object class whenever they need to interact with the UI of that page, the benefit is that if the UI changes for the page, the tests themselves don't need to be changed, only the code within the page object needs to change.

Advantages of Page Object Model:

- Code reusability – We could achieve code reusability by writing the code once and use it in different tests.
- Code maintainability – There is a clean separation between test code and page specific code such as locators and layout which becomes very easy to maintain code. Code changes only on Page Object Classes when a UI change occurs. It enhances test maintenance and reduces code duplication.
- Object Repository – Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.
- Readability – Improves readability due to clean separation between test code and page specific code.

5. What is difference between POM and PageFactory?

A Page Object Model is a test design pattern which says organize page objects as per pages in such a way that scripts and page objects can be differentiated easily. A Page Factory is one way of implementing Page Object Model which is inbuilt in selenium.

In plain POM, you define locators using 'By' while in Page Factory, you use @FindBy annotation to define page objects.

Page Object Model is a design approach while PageFactory is a class which provides implementation of Page Object Model design approach.

In plain page object model, you need to initialize every page object individually otherwise you will encounter NullPointerException while In PageFactory all page objects are initialized (Lazily) by using initElements() method.

6. What is difference between finding element using a method vs using findBy annotation?

FindElement:
This is one of the abstract methods , should be used to find the single web element and return type is single web element.

@Findby:
This is one Annotation in Page factory model and used to locate one or more WebElements using a single criterion.

7. How you conducted Batch Testing in your project?

When we run multiple script, with Automation Tool, that is called batch testing.
We conducted Batch Testing using **TestNG** Testing Framework using **@groups**.
In **Cucumber** we are using the **@Tags** options for running the batch of test cases.

8. What drivers do you use in your framework?

ChromeDriver
InternetExplorerDriver
GeckoDriver

Mostly we perform your execution in the ie and chrome browser

9. How would you handle test data in your framework? How do you run tests using excel data ?

Cucumber framework:
Using Scenario Outline (Inbuilt mechanism to retrieve data), DataTable (Wrote logic to retrieve data in a form of List<Map>), and Excel (Reading using Apache POI and for loop)

TestNG:
Using Excel (Apache POI and DataProvider)

Please make sure you know how to work with Excel

10. Tell me about your reporting systems? Where do you store the test results/reports?

We are using Cucumber Reports/TestNG reports
Reporting plays a key role in any project at the end of the day everyone is interested to see your reports not the code.

Cucumber provides default HTML report and in my project we are using Maven Cucumber plugin which provides following benefits:

- Graphical representation: It will convey the statistical results like % passed / failure status
- Consolidated view: Detailed execution status in numbers will be displayed.
- Error Analysis: It will help the user to analyze and debug the issue. User can access the test data and Test evidence from reports, which are the most important aspects while fixing the bugs.

TestNG Reports

TestNG library brings a very convenient reporting feature. Once we execute the tests, TestNG generates a test output folder at the root of the project.It combines two kinds of reports.

- Detailed Report - report in the <index.html> file. It combines the detailed information like the errors, test groups, execution time, step-by-step logs and TestNG XML file.
- Summary Report - it is the trimmed version and informs about the no. of “Passed”/”Failed”/”Skipped” cases. You can see it from the <emailable-report.html> file.
- Extend Report - Extent Report is an HTML reporting library for Selenium WebDriver for Java.It is a great tool we can use within our automation framework to make excellent execution reports.

11. What type of tests have you automated? When do you do automation in your sprint? How many test cases you have automated per day?

Mostly I have automated functional regression test (includes smoke sanity and regression, database and API testing) and we do in sprint automation of user stories. Also in my previous projects when I was only part of the automation team we were doing automation after sprint was over. **Ask them how they do automation in their company.**

There is no right or wrong answer on the automation of test cases per day:

- It is all depends on the functionality and complexity of the scenario.
- I always follow coding standards and focus on the quality of my scripts not quantity. Sometimes just find a right locator and specify dynamic xpath takes time
- Since I work in Agile environment some days will be spending attending sprint meetings.
- When developer fix any bugs I jump on retesting that bug based on its priority and severity. On top of that when some of the team members have any challenges and need my help I always try to help them.

12. What is smoke testing and how do you run it in your project? How many test cases are there and how often do you perform smoke testing?

Smoke Test is a process where the software build is deployed to QA environment and is verified to ensure the stability of the application. All the critical functionalities would be put to test here. Once the build clears the first state, it will be taken for further QA tests, and if the build fails the test, it would be rejected and QA team would revert it back to a previously accepted version or reject it completely.

A major benefit of running a smoke test is that it provides quick feedback within a matter of few minutes and testers don't have to wait hours to get results. In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing.

We have a dedicated pipeline for our smoke test within our Jenkins and since Jenkins is a web based application, anyone can access it by using its URL. We provide following arguments during the initial configuration:

- Environment in which we need do the smoke test
- Browser: IE is default
- Suite: smoke test. Suite

And then whenever there is a code push our smoke test pipeline was automatically triggered.
Runner.exe also creates a nice result file with the test cases details execution. Jenkins sends the smoke test result to the entire team members and in case of issues, developers can look into that immediately.

13. What is regression testing and how do you run it in your project? How many test cases are there and how often do you perform regression testing?

Regression testing is a type of software testing where testers verify whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.
When we are going performed Regression testing:

- Bug fix
- New feature added
- Changes in existing feature
- Performance issue fix

Main purpose of regression testing is to check the stability of the existing functionality with new features being implemented in a sprint. Regression testing is crucial stage for the product and is very useful for the developers to identify the stability of the product with respect to changes or new requirements.
From Automation perspective, Automation of regression Test Cases is the highest priority in the projects. Regression testing is a very heavy process which requires a lot of effort, time and preparation. Currently in my project we have almost 900 test cases in our regression suite from which close to 75%-80% are automated. As we follow agile, our releases scheduled after every other sprint.
For Cucumber: whenever we write test cases in our future files we add @Regression tags for each scenario and later through the Regression runner class we execute those tagged scenarios.
For TestNG: we have regression testng XML in that we add regression related test cases. Regression also been triggered from Jenkins.

Meanwhile, I also perform manual test cases execution and on top of that I always try to perform an ad hoc testing.
Once execution is done, I will analyze results for any failed test cases. I will identify whether test cases fail due to environment performance issues, script related issue or application issues. Once I know that script failed due to the application issues I will retest that particular test cases and try to reproduce the bug and verify it against requirements. Once I confirm the bug I will logg it into the Jira application providing description of the bug with severity and priority level, steps to recreate and appropriate screenshots.

14. What is Automation Testing? What are the benefits of Automation Testing?

Automation Testing means using an automation tool to execute your test case suite. The automation software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports.

Faster Feedback: Automated testing comes as a relief for validation during various phases of a software project. This improves communication among coders, designers and product owners, and allows potential glitches to be immediately rectified. Automated testing assures higher efficiency of the development team.

Accelerated Results: Owing to the quick implementation of automated testing, plenty of time is saved even for intricate and enormous systems. This allows for the testing to be carried out repeatedly, delivering faster results each time with lesser effort and time.

Reduced Business Expenses: While the initial investment may be on the higher side, automated testing saves companies many a penny. This is predominantly due to the sharp drop in the amount of time required to run tests. It contributes to a higher quality of work, thereby decreasing the necessity for fixing glitches after release and reduces project costs.

Testing Efficiency Improvement: Testing takes up a significant portion of the overall application development lifecycle. This goes to show that even the slightest improvement of the overall efficiency can make an enormous difference to the overall timeframe of the project. Although the setup time takes longer initially, automated tests eventually take up significantly lesser amount of time. They can be run virtually unattended, leaving the results to be monitored towards the end of the process.

Higher Overall Test Coverage: Through the implementation of automated tests, more number of tests can be executed pertaining to an application. This leads to a higher coverage that in a manual testing approach, would imply a massive team limited heavily with their amount of time. An increased test coverage leads to testing more features and a higher quality of application.

Reusability of Automated Tests: Due to the repetitive nature of test automation test cases, in addition to the relatively easy configuration of their setup, software developers have the opportunity to assess program reaction. Automated test cases are reusable and can hence be utilized through different approaches.

15. Why is manual testing still exists, even though automation has been proven to be more efficient?

1. When a new feature is implemented, initially running a manual test is essential to ensure functionality meets acceptance criteria. In order for a test script to be properly automated, the exact test steps need to be manually executed and documented.

2. Manual testing also includes exploratory testing which enables testers to find

weaknesses, and risks. This enables testers to determine which part of the application needs more testing. Manual testing is less expensive in the short-term. If you're only testing a simple application once, and don't expect lots of updates, manual testing doesn't require you to invest in expensive tools or software.This is considered flexible and on-the-fly testing. Automation testing requires you to write, program, and review test cases for your software. But, if you really only need to test one small change, you can manually test it faster than it would take to automate. This is beneficial when a high priority bug fix needs to go out to production immediately.

3. Manual testing provides the human element to make sure the feature is user-friendly. A manual tester could tell you that the contrast between a button and the background is too light, which makes it hard to see the button and understand what action needs to be taken. This type of user interface (UI) feedback is something automated testing wouldn't find, making manual testing closer to the sort of feedback you might hear from actual customers.
Manual testing is valuable early in the development of features and user interfaces because layout and controls are often changing (almost daily) in response to design considerations and user feedback.
Being able to predict what your users will or won't like—things a computer can't give feedback on—ahead of time can influence your design and make it better from the bottom up.

4. Manual testing identifies bugs that can be found in areas users wouldn't expect. Typically, these bugs are found by quality assurance teams who were looking for something other than the issue they found. Automation will not find errors it wasn't scripted to find. It's impossible to catch all bugs on the first round of testing, so manual testing is a faster "catch-all" for what could possibly be missed.

5. Manual testing is used to reproduce customer-found defects. This enables the quality assurance engineer to duplicate the steps and provide detailed information to the developer about browser related issues or data issues that might not be found by automation.

6. Often, in an agile environment, things are changing and features are being enhanced. This requires constant rewrites to the automation scripts and creates additional tasks for the team. Manual testing does not need increased work to be able to adapt to any new changes.

Automation testing is a benefit to any company, but manual testing should never be replaced. Manual testing is always going to find things automation won't find.

16. If I hire you and ask you to come and start automation, how would you do it?

- Analyze application
- Choose and Find Automation tool experts
- Create and Execute Manual Test Cases
- Analyzing test cases to determine those which are best suited for automation
- Create Smoke Test and present to the team
- Creating the test automation Framework
- Writing Scripts
- Reporting
- Maintenance of Scripts

Automation Life Cycle:

Simply we can say as:
Analyze the Application -> Select the tool -> Identify the scenarios -> Design test scripts -> Modify test scripts -> Run the test scripts -> Report defects

IN detail explanation:

Automation Feasibility Study

Before kicking off implementing test automation, it is mandatory to analyze the feasibility of the application under test (AUT). Whether AUT is a right candidate or not for the test automation?

Main focus

- Which test case/module can be automated and how we can automate them.
- Which tools we can use for the application (like Selenium,QTP etc) and which tools will be best of our application
- Consideration like Team size, effort and cost involved for tools which we will use.

Test Planning/ Test Design

This phase defines approach and any risks
Following are the Test automation Framework available

- Cucumber framework
- TestNG framework
- Data driven framework
- Hybrid Framework (preferred method)

Then

- Schedule
- Number of resources
- Mode of communication process
- Defining in-scope and out-of-scope

Environment Setup/Test lab setup

In this phase whatever tools you need you will start setting it up

Test Script development/ Automation test case development

In this phase we have to start developing automation script and make sure all test script are running fine and should be stable enough.

- Start creating test script based on your requirement
- Create some common method or function that we can reuse throughout your script
- Make script easy, reusable,well structured and well documented so if third person check the script then he/she can understand scripts easily.
- Use better reporting so in case of failing we can trace our code

Test script execution

In this phase we have to execute all your test script.
Some points to remember while execution:

- Script should cover all the functional requirement as per test case.
- Script should be stable so it should run in multiple environments and multiple browsers (depends on your requirement)

- Can do batch execution also if possible so it will save time and effort.
- In case of failure your script should take screenshots.

If test case is failing due to functionality, we have to raise a bug/defect.

Generate test result / Analyses of result

Last phase of Automation test life cycle in which we gather test results and share with team/client/stakeholders:

- Result analysis, how much time automation takes
- Report generation, Cucumber Maven or Extend Reports
- Documenting the issues and knowledge gained
- Preparation of team/client/stakeholders presentation