What is Maven : Maven is Build tool, which helps developer in building their Projects.
A build tool is a tool that automates everything related to building the software project.
Pom.xml file
Need => we develop business logic + third party application + third party project + download / import project

What is Selenium Framework? Types of Framework?
The **Selenium Framework** is a code structure that makes code maintenance easy and efficient. Without frameworks, users may place the “code” and “data” at the same location which is neither reusable nor readable. Frameworks produce beneficial outcomes like increased code reusability, higher portability, reduced cost of script maintenance, better code readability, etc.
There are mainly three type of frameworks created by Selenium WebDriver to automate manual test cases

- Data Driven Test Framework
- Keyword Driven Test Framework
- Hybrid Test Framework

Data Driven Framework in Selenium
Data Driven Framework in Selenium is a method of separating data sets from the test case. Once the data sets are separated from the test case, it can be easily modified for a specific functionality without changing the code. It is used to fetch test cases and suites from external files like excel, csv, .xml or some database tables.

Keyword Driven Framework in Selenium
Keyword Driven Framework in Selenium is a method used for speeding up automated testing by separating keywords for common set of functions and instructions. All the operations and instructions to be performed are written in some external file like an Excel sheet. Users can easily control and specify the functionalities they want to test.

Hybrid Framework
Hybrid Framework in Selenium is a concept where we are using the advantage of both Keyword driven framework as well as Data driven framework. It is an easy to use framework which allows manual testers to create test cases by just looking at the keywords, test data and object repository without coding in the framework. BDD (feature file)

POM(Page object module)

Page Object Model (POM) is a design pattern, popularly used in test automation that creates Object Repository for web UI elements. The advantage of the model is that it reduces code duplication and improves test maintenance.

Under this model, for each web page in the application, there should be a corresponding Page Class. This Page class will identify the WebElements of that web page and also contains Page methods which perform operations on those WebElements. Name of these methods should be given as per the task they are performing, i.e., if a loader is waiting for the payment gateway to appear, POM method name can be waitForPaymentScreenDisplay().

create a separate class file which would find web elements, fill them or verify them. This class can be reused in all the scripts using that element. In future, if there is a change in the web element, we need to make the change in just 1 class file and not 10 different scripts

This approach is called Page Object Model in Selenium. It helps make the code more readable, maintainable, and reusable

Advantages of POM

Page Object Design Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.

The Second benefit is the object repository is independent of test cases, so we can use the same object repository for a different purpose with different tools. For example, we can integrate Page Object Model in Selenium with TestNG/JUnit for functional Testing and at the same time with JBehave/Cucumber for acceptance testing.

Code becomes less and optimized because of the reusable page methods in the POM classes.

Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

Page Factory in Selenium is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized. It is used for initialization of Page objects or to instantiate the Page object itself. It is also used to initialize Page class elements without using "FindElement/s." Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of class PageFactory in Selenium, we use annotations @FindBy to find WebElement. We use initElements method to initialize web elements

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end to end testing easy.
Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.
There are six major advantages of TestNG over JUnit:

- Annotations are easier to understand
- Test cases can be grouped more easily
- Parallel testing is possible
- All test case will execute
- Generate report
- Execute only failed test classes in new build

Annotations
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.
@BeforeSuite: The annotated method will be run before all tests in this suite have run.
@AfterSuite: The annotated method will be run after all tests in this suite have run.
@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.
@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the 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.
@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.
@AfterClass: The annotated method will be run after all the test methods in the current class have been run.
@BeforeMethod: The annotated method will be run before each test method.
@AfterMethod: The annotated method will be run after each test method.
@Test: The annotated method is a part of a test case

Sequence Assignment

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

Keywords in framework

1. **dependsOnMethod**

Test Dependency Order : Dependency is a feature in TestNG that allows a test method to depend on a single or a group of test methods.

TestNG allows you to specify dependencies with: Using attribute dependsOnMethods in @Test annotations.

Important Notes on Test Dependency:

The dependency on multiple test methods are configured for a test method by providing the comma separated dependent test method names.

If a dependent method fails, all the subsequent test methods will be skipped, NOT failed.

2. **Priority**

In automation, many times we are required to configure our test suite to run test methods in a specific order or we have to give precedence to certain test methods over others.

TestNG allows us to handle scenarios like these by providing a priority attribute within @Test annotation. By setting the value of this priority attribute we can order the test methods as per our need.

if we have not given any priority then by default priority is zero

@Test(priority=1)

3. **enabled**

If the enabled=true then the test will execute and if enabled=false then the test will NOT execute.

@Test(enabled=false)

4. group

Test Grouping : Grouping is a feature through which you can easily categorise your test cases based on requirement.

User can assign any Group to TestCase by passing groups name to @Test Annotation.

@Test(groups={"groupname"})

User can maintain the execution of Groups via testng.xml file.

After grouping in Selenium we can specify the include and exclude in testng.xml.

<include> – It tells testng.xml that which group we need to execute.

<exclude>- It tells testng.xml which group we have to Skip

Module1 :- Test, Working fine, Prod

Module2 :- Test, Working fine, Dev- Some of TC from module1 need to be tested again (Impact of module 2)

Module 3 :- Sanity

Test1 :- Regression

Test2

Test3 :- Regression

Test4 :- Regression

.

.

Test100

5. invocation count

TestNG supports multi-invocation of a test method, i.e., a @Test method can be invoked multiple times sequentially or in parallel. If we want to run single @Test 10 times at a single thread, then invocationCount can be used.

@Test(invocationCount = 10)

6. Timeout

The automation test suites have the tendency to take too much time in case the elements are not readily available for interaction. Also, in certain tests, we might have to wait for some asynchronous event to occur in order to proceed with the test execution.

In these cases, we may want to limit the test execution time by specifying an upper limit of timeout exceeding which the test method is marked as a failure. TestNG provides us timeOut attributes for handling these requirements.
@Test(timeOut = 1000)

Assertion
Assertion : Assertion is performed to compare the Actual result with expected result.
Assert helps us to verify the conditions of the test and decide whether the test has failed or passed. Actual result is compared with expected result with the help of Assertion. It means verification is done to check if the state of the application is the same to what we are expecting or not. For creating assertion we are going to use the Assert class provided by TestNG.

Types of Assertion
Hard Assertion
Hard Assert throws an AssertException immediately when an assert statement fails and test suite continues with next @Test
The disadvantage of Hard Assert – It marks method as fail if assert condition gets failed and the remaining statements inside the method will be **aborted**. To overcome this we need to use Soft Assert

Assertions provided by the TestNG.
assertEqual(String actual,String expected)
assertEqual(String actual,String expected, String message)
assertEquals(boolean actual, boolean expected)
This is used to compare expected and actual values in the selenium webdriver. The assertion passes with no exception whenever the expected and actual values are same. But, if the actual and expected values are not same then assert fails with an exception and the test is marked as failed.

Assert.assertEquals(actual, expected);
assertTrue(condition)
assertTrue(condition, message)
Assert.assertTrue(rv.isDisplayed)
Assert.assertTrue(rv.isSelected)
Assert.assertTrue(rv.isMultiple)

This type of assertion is used when you are checking if condition is true. That is when we are dealing with boolean values this assertion is used. Whenever test case passes it returns true and if condition is false then it skips the current method and jumps to next.

assertFalse(condition)

assertFalse(condition, message)

assertNull –

This assertion checks if the object is null or not. It aborts the test if object is null and gives an exception.

Syntax :

Assert.assertNull(object);

assertNotNull –

Assert.assertNull(str); => fail

This assertion checks if object is null or not. It aborts the test if object is not null that is if object is having any value and gives an exception.

Syntax :

Assert.assertNotNull(object);

Program:

Soft Assertion

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.

If there is any exception and you want to throw it then you need to use assertAll() method as a last statement in the @Test and test suite again continue with next @Test as it is.

We need to create an object to use SoftAssert which is not needed in Hard Assert.

Program:

Difference between testNG and jUnit

testNG	jUnit
TestNG is a testing framework that was developed by Cédric Beust.	JUnit was developed by Kent Beck, David Saff, Erich Gamma, and Kris Vasudevan.
TestNG is a Java-based framework that is an upgraded option for running tests.	JUnit is an open-source framework used to trigger and write tests.
TestNG can run parallel tests.	JUnit does not support to run parallel tests
It supports advanced annotation	It does not support advanced annotation.
Dependency tests are present in TestNG	The dependency tests are missing in JUnit
Tests can be grouped	Grouping tests is not possible in JUnit.
Writing tests & configuring is easy in TestNG.	Running tests need a dependency on JUnit

Failed xml file

Test Suite:

Select the test classes needs to be run -> Right click -> TestNG -> Convert to TestNG

Parallel Execution.

OOPS concepts in selenium:

Inheritance :- TestBase extend to page classes and test classes

Abstraction :- In Test class we call method so the execution code is hidden from test cases

Constructor :- Page class to initialise the object/elements of web pages

Interface :- Itestresult, Webdriver, takescreenshot, WebElement

Encapsulation :- Page class, object repository

Polymorphism (overloading, **overriding**) :- Assertion

This keyword

Scrum master :- BA/TL/PM/PO/Sr. Team Member

Stand up / Scrum call – 15 min (14) – 30min. 9 am -9.30 am

Update :-

Good Morning everyone / Team !

What we did? / what we are going to do / Blocker? TA-15390.

Bye Team, have a nice day

Bye Team have a happy weekend

Start working as per ticket assigned

Communication is happen on Teams / Hangout – Channels / Groups (Client group, **Working Team group**, Team with Higher Authority group)

If any issue occurred while working

Hello Team, I have issue with QA4 env and its not working on my side :- Working Team group

If problem really exist then post on client group or asked specific person to do it like

@Andy please look into this matter

And mention that issue in next day scrum call

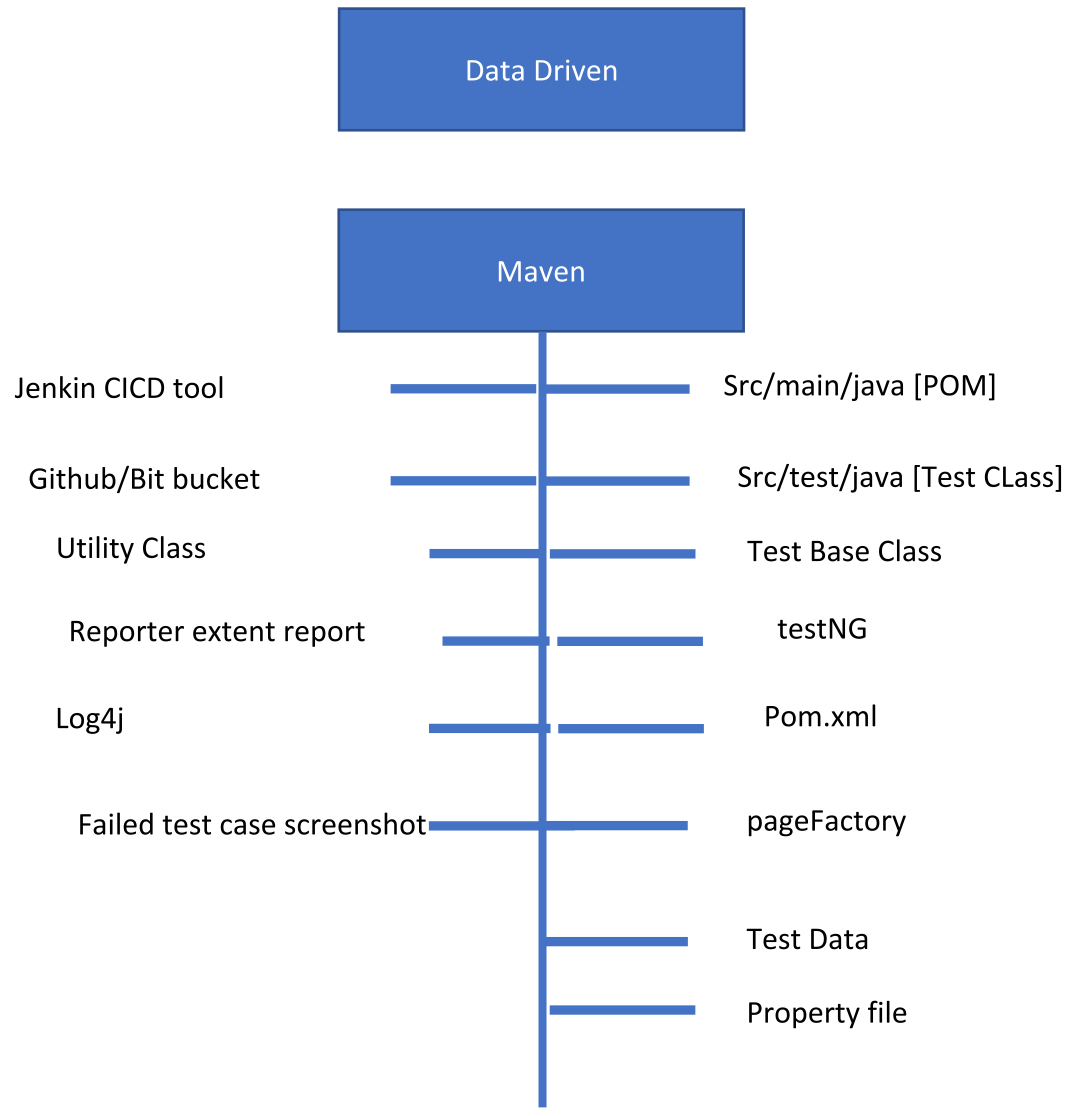
QA1 – QA5

Automation life cycle

Test plan -> KT to automation tester by manual -> setup test environment or feasibility check -> develop basic script -> enhance basic script -> execute (report bugs) -> generate report

Push the code to git

How to explain framework



PageFactory :- all elements of class initialize with driver using this keyword

Types of framework

Selenium framework is a code structure for making

- Code maintenance simpler
- Code readability better

A framework involves breaking entire code in smaller pieces of code which test a particular functionality

The code is structured such that the Test Data is separated from actual test cases which will not test the functionality of web application

It can also be structured in a way where the test cases which need to be executed are called from external source like excel sheet or csv file

Benefits of selenium framework

- Increase code reusability
- Increase code reusage
- Higher portability
- Reduce script maintenance

Data driven framework

- It is technique of separating test data with actual test cases
- It depend on input test data
- The test data is fed from external source such as excel or csv file

Since test code separated from test data, we can easily modify test cases of particular functionality without making any changes to code
Ex. if we want to modify code for login functionality then you can modify login instead of modifying any other dependent portion in same code.

Beside that we can easily control how many data need to be tested. We can easily increase number of test parameter by adding more username and password field to excel file

POM
It is java design pattern use to design classes in test script
It follows encapsulation concept

- Data member should be declare globally with an access level private
- Initialize within constructor with an access level public
- Utilize within the method with access level public

Note :- Number of data member that need to be created under POM classes depends on the number of component that need to be handled in web page

POM class does not contain main method, we require another class i.e. test class
Test class contain all navigation steps to test an application

POM classes

- It depend on web page present in the application
- For each page POM class is created (page = class)
- Data member in POM class are created by data encapsulation by using pageFactory
- Number of data member present in POM class depends on number of web page element
- Each declare data member should be initialize and utilize within POM class

Test Classes

- It depend on number of test cases written by manual test engineer
- Test case contain navigation steps and input that need to be given to component
- In test class data or input can be given directly or through external sources like excel

Page Factory

It contain static method initElements :- to initialize data member in page factory we need to use initElements method in constructor
pageFactory.initElements(arg1,arg2);
initElement initialize data member by identifying each component in web page by using @FindBy annotation which takes locator type as an argument

@FindBy (locator type = “path”) private webElement data_member;

Working

While executing test script initElements method convert all data member @FindBy annotation to findElements this process is known as basic initialization or early initialization
@FindBy (id = “path”) private webElement username;
username = driver.findElement(By.id(“path”));

- To perform action on component we need to call methods
- Before performing each action init element method will identity component present or not then it will do complete initialization
- This process is known as late or lazy initialization

Difference between page factory & POM

POM	**Page Factory**
It initialize all data member present in class completely while performing action on component	It initialize data member present in class before performing each action
It use when web page does not contain hidden component	It can be use when web page contain hidden element