How To Explain Test Automation Framework To The Interviewer

We need to specify in and out of our Test Automation Framework such as programming **language** used, **Type of framework** used, Test Base Class (Initializing WebDriver, Implicit Waits), How we separate Element locators and tests (Page Objects, Page Factory), Utility functions file, Property files, TestNG annotations, How we parameterize tests using Excel files, How we capture error screenshots, Generating reports(Extent Reports), Emailing reports, Version Control System used and Continues Integration Tool used.

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 Data-driven Framework by using Page Object Model design pattern with Page Factory.

POM: 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.

Test Base Class: Test Base class (TestBase.java) 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 (AKA Functions Class): Utility class (TestUtil.java) 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.

*Roles & Responsibilities of automation Tester

1. Selenium Environment setup : download & install eclipse, jav lang, Configrution Selenium jar files TESING, Maven project etc.
2. Inspeet elements / object: using firepath

3. Creating test case using element locators and Selenium webdriver Command
    - element locater for identifying element
    - Selenium webdriver for performing operation on elements.

4. Enhancing Test Cases & using prg. Feature

flow Control statements ,exception handling ,Conditional (Selenium SUPP 6 lang)

adding Comment, error handling verification etc

5. Grouping Test cases, prioritizing test cases, executing test batches & generating test reports Using testing framework

6. Data driven testing : DB testing, cross browser testing executing same functionality with multiple set of Data

7. Analysing test result & reporting defects.

8. Selecting test cases for regression testing, defect tracking

9. Regression testing on modify builds

10. Final regression testing

11. Maintance of test automation resourses

Steps Involved In Automation

1] Selecting The Test Tool

2]Define The Scope Of Automation

3]Planning Design & Devl

4]Test Execution

5]Maintance

Steps Involved In Planning Phase Of Automation

1] Selecting the Right Automation Tool

2] Selecting Automation Framework If Any

3] List of Scope for Automation Test Environment

4] Preparing Grant Chart for Dev. And Executions

5] Identify the Test Deliverables

The conditions where we can't use automation in agile

1] When agile testing always asks for changes or stack holder changes in requirements.

2] When exhaustive level of documentation is required in agile

Primary features of good automation tool

1] Test envoirment support & easy to use

2] Good debugging facility

3] Rebust object identification

4] Object & image testing ability

5] Testing of Database

6] Supports multiple frameworks

List of Challenges faced in project.

1) Domain knowledge of application
2) vast application, flows, functionalities & module integration was
3) use of dynamic Xpath
4) Handling multiple exceptions while test execution. Eg Stale Element Exception, NullPointer Exception
5) Dependencies of test Cases on one another
6) Execution of simple text Case work but files in Suite
7) use of Before methed implemented to minimize dependencies
8) maintenance of test Data & pre conditions on multiple
9) multiple unexpected. Pop up handling.
10) Application freeze because of multiple user using single part.
11) many setting related test cases were tested an different part to minimize the effort on other test cases