TELL ME ABOUT YOUR FRAMEWORK

1. **Tell me about your framework?**

Version 1

- In my last project, in order to design my framework, I used different management and automation tools as **Eclipse IDE** / **IntelliJ**, **Selenium WebDriver** for **browser automation**, **Maven** for dependencies, **Cucumber** and **Jenkins**. I also used **POM** structure in order to keep my code organized and clean. So, I basically 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 have separate classes where I keep my implemented **step definitions**. Also, for each scenario(positive or negative) I created **Cucumber feature files** where I used **GHERKIN** language in order to describe my **test scenarios**, by doing that I'm making sure that my test cases are understandable for each member of my team, even for those who aren't technically strong. I created another separate package for my utilities where I would store all my drivers, page files or utilities that I could possibly run. Reporting is done in cucumber and Jenkins. Actually, Jenkins is using Json file generated by the cucumber each time the test is run in Jenkins. Of course, all my code is stored in GIT so I can share it with my team members when necessary. Last thing that I would like to add is that I'm using Jenkins to run my smoke and regression tests by taking the code from GIT(Version Control).

Version 2

- I manage my dependencies and **plug-ins in pom.xml**, running my tests from command line. It is written in **Java**. I use **Eclipse IDE /IntelliJ** in my company. My framework uses **Selenium WebDriver** for **browser automation**, and **JUnit** for starting my tests and for assertions, My framework is **BDD**, which uses **Cucumber** to write tests in feature file, organize test suites like @smoke, @api. Our feature files are written in **Gherkin** language to make it easy to understand for non-technical people. My framework generates **HTML** reports with exact steps from the feature file.
  In my framework; Other than feature files, I have a runner class that runs my tests and helps to generate codes for step definition from my feature files. It also contains locations that show where my feature files and **StepDefs** (Glue) are. Step definition classes where I have my methods to execute feature files, Driver class is designed as **Singleton**. Configuration Reader and properties file. I also have page classes where I locate my elements for each page. Besides these, I have utility pages where I store my **reusable codes**. Lastly, I have hook class that implements my codes which run before and after all my tests this is where I invoke my **TakeScreenShot** for each failure. My framework supports **Data-Driven testing** using **Apache POI**, **Scenario Outline, excel and csv files.**
  In my framework, I can also perform Database testing through JDBC driver. I am using **JIRA** as project management and bug tracking tool. For Continues Integration and test scheduling, I use **Jenkins**. And I use **Selenium Grid** for multi browser parallel testing.

2. **Summary of my framework?**

FRAMEWORK

Data Driven Framework

a framework where tests are executed based on set of data, framework is designed to read data from outside sources like excel and run tests based on the data. In data driven framework we can execute the same test multiple times against different sets of data.

Page Object Model Framework

uses the page object design pattern according to which we create a separate java class for each page of the application

Behavior Driven Framework

Keyword Driven Framework

In KDF we use keywords in outside source (excel. csv) to run our tests. Framework is designed to read the data and steps from excel and execute actions based on it.
Once KDF is set up even non-technical testers can write and execute automated tests.

Hybrid Framework
hybrid framework is a framework uses at least two of the types given above

TOOLS

Java: My framework is written using Java language

Maven: My framework created as a maven project, maven is used to manage dependencies and also run our tests as mvn goals from terminal

Selenium WebDriver: a library/tool/api which is used to automate the browser, it interacts with the browser.

TestNG: used to group tests using xml files, do soft and hard assertions, create test methods, run in certain order
Extent: my framework generates detailed HTML reports with is easy to read and understand to non-technical team members. My reports have details test steps and screenshots for any failures that may occur. It can also do metrics on what percentage is passing, failing, skipped etc.

IDE: I use IntelliJ in my current framework, but I am also quite comfortable with Eclipse with I used previously

DESIGN

Page Object model: my framework used page object model according to which I created a separate class for the pages of my application.

PageFactory design: a design which makes it easy to access the page object class.
Below not Page factory design. it is class which has the same name as the PageFactory design:

```
PageFactory.initElements(driver, this)
```

Singleton Driver: My frameworks uses a singleton pattern to share the webdriver instance between different classes

TestBase: My framework has a testbase class which my tests extent. testbase class has the common steps for all my tests.

Configuration.feature file: used to store the important test data

Utilities: have reusable utilities which can be used across different classes of my framework

Benefits:

Easy to maintain:

- My framework uses page object model which makes it easy to maintain. for example, if I have to update any locator, I only need to do one code change.
- I try to make my tests independent from each other. this mean if I update one test, it will not affect others and also if one fails, others will not be affected.

Easy to extend:

- it is easy to add new test cases to my framework.

Easy to reuse:

- I have page object model, utilities which I can reuse for any tests.

Multi browser testing: My framework can run the same tests against different browsers with minimal code change. Junit, TestNG are have their methods for multi-browser testing. Except them we use Selenium Grid

Types of tests: My framework can test the UI, database and API of the application.

Packaging: I have create different packages for different types of classes and logic. Each page package only contains classes with same functionality.

Naming conventions: in my team we pay a lot of attention to coding standards, especially naming conventions. Classes, methods variable are named on based on what they do and follow a standard

Page object class:
homepage, loginpage
variable:
loginButton, signOutLink
methods:
login(): these methods only used to login, not for other functionality.

3. Product Structure? Web Application Structure

- Front End
  - **SmartClient** Framework → it is a **JavaScript (Js)** UI Framework from a San Francisco based client. It is 100% Microsoft compatible
  - Terminology
    - **HTML structure** : is only responsible data
    - **CSS presentation/appearance** : for design
    - **JavaScript dynamism/action** : for function
- Back End → **ServerSide Project**
  - We have so many **prods** (machines) all around the world. Especially in USA, Australia, Singapore, France
  - We open three servers in each machine
    - **Application Server**
      - **IIS Application** → The architecture where the code runs -handle tasks. We give public URL addresses for each client. Backend and Frontend projects run on this Application servers by the **IIS Application**.
      - Our test framework is also hosted in this server.
      - Our Developer team uses **C++** for developing the applications and we testers use **JAVA** for Testing project
    - **Report Server**
      - This is a service of **Windows SQL Service**.
      - We have Report services which creates and calls Reports from UI.
    - **Microsoft SQL Server** → Our SQL tables are in these servers. We host data on Microsoft SQL Server.

4. Mobile Application Structure

- **Android Apps** → We develop the applications on Android Studio. And we use many AWS services such as Push Notification, Messaging, and Emailing
- **IOS Apps** → We develop on XCode 9
- **FireBase** → To track Mobile Apps usage, bugs and crashes (Crashlytics)
- **Appium** → To test mobile apps (we have just started)
- **Selendroid** →We are observing and considering using.