POM

1. **Language**: In our Selenium Project we are using **Java language.**
2. **Type of Framework**: In our project, we are Using **Behavioural-driven Framework** by using Page Object Model design pattern with page Factory.
3. **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.
4. **Packages**: We have separate packages for **pages and Tests**. All the web page related classes come under **Pages** package and all the tests related classes come under **Tests** package.
5. **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.
6. **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 screenshorts, accessing excels, sending email** etc.,) which can be commonly used across the entire framework. The reason behind creating utility class is to achieve reusability. This class extends the TestBase class to inherit the properties of TestBase in TestUtil.
7. **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 easy to maintain.
8. **Screenshots**: Screenshots will be captured and stored in a separated folder and the screenshots of a failed test cases will be added in the extend reports.
9. **Test Data**: All the historical **test data will be kept in excel sheet** (controller.xlsx). By using 'controller.xlsx', we pass test data and handled data driven testing. We use Apache POI to handle excel sheets.
10. **TestNG**: Using TestNG for **Assertions, Grouping and parallel execution**.
11. **BDD framework i**.e. Behaviour driven Development is a software development approach that allows the tester/business analyst to create test cases in simple text language (English). The simple language used in the scenarios helps even non-technical team members to understand what is going on in the software project.
12. **Maven**: Using Maven for **build, execution, and dependency purpose.** Integrating the TestNG dependency in POM.xml file and running this POM.xml file using Jankins.
13. **Version Control Tool**: We use **1** as a repository to store our test scripts.
14. **Jenkins**: By using Jenkins CI (Continuous Integration) Tool, we execute test cases on daily basis and for nightly execution based on the schedule. Test Result will be sent to the peers using Jenkins.
15. **Extend Reports**: For the reporting purpose, we are using Extent Reports. It generates beautiful HTML reports. We use the extent reports for maintaining logs and to include the screenshots of failed test cases in the Extent Report.