**EXPLANATION OF FRAMEWORK**

* In our project, we have implemented a hybrid framework that combines both data-driven and modular-driven frameworks. We have chosen Java as our programming language, Selenium as the automation tool, and TestNG for unit testing within a Maven project. To manage our source code, we use GitHub, and for development and deployment automation, we rely on Jenkins.
* A base class for pre and post conditions.

A database utility for reading data from a database.

A property utility for reading property files.

An Excel utility for reading data from Excel files.

A WebDriver utility that implements methods such as those for the JavaScript executor, Actions class and the Select class.

An ITestListener utility for generating reports.

A Java utility for generating random numbers.

A TakeScreenshot utility to capture screenshots of web elements and web pages.

* In addition, we have an interface called IAutoConstants, which stores file paths and screenshot-related constants. In addition, we use an interface called IAutoConstants. This ensures that file paths and constants related to screenshots are used consistently throughout the framework. They are versatile tools that streamline and enhance the functionality of our framework, making it more efficient and effective in handling diverse tasks.
* This "Generic Utility" phase sets the groundwork for the "Object Repository" phase, where we create object repositories to make it easier to work with different parts of our application.
* In the "Test Scripts" phase, we use the utility classes and object repositories to write the actual test scripts. These scripts are then grouped into "Testing Suites" with .xml file extensions. The "Maven Execution" phase handles running these test suites by using Maven.
* For managing our source code and collaborating, we use the "Source Code Management" phase. This is where we push our code to GitHub and perform activities like pulling, branching, and merging to keep everything organized.
* Next, in the "Jenkins Integration" phase, we set up Jenkins to execute our tests. This can be done manually, on a schedule, or automatically when there are changes in our source code, thanks to Jenkins' "Poll SCM" feature.
* Reporting is an important aspect of our framework, and in the "Reporting" phase, Jenkins generates detailed reports and takes screenshots to provide a clear view of our test results.

**Frame Work Flow:**

2.Object Repository



4.driver.xml/testng.xml/suite

LoginPage.java

CartPage.java



Profiling

Execution xml File

Batch.xml

Group.xml

Parallel.xml

CBT.xml





bsxaaxsz 

Pushing

Pulling

Branching

Merging

TC01\_Test.java

TC02\_Test.java

9.Screenshot

6.SCM(Github)

3.TestScripts