MY FRAMEWORK:

- As per the **norms of Automation**, we should always **softcode**, so designing a hybrid framework is one of the important aspects in any project.
- As I said earlier, we were following **Agile-Scrum methodology.** I was involved in developing a **Hybrid Framework** which includes **Data-driven, Keyword-driven, POM classes, TestNG** and **Modular driven frameworks** as well.
- For all the **WebDriver** related actions, we used to maintain a **WebDriver Utility file**.
- For all the **reusable methods**, we used to maintain a **Generic Library file.**
- Fortunately, I had a chance to add a few **reusable methods** in it.
- Since few things are **not going to be changed** during the entire project, I used to store all the **common paths** and **user directories** by creating an **Interface** called **IPathConstants** wherein all the variables are **by default**, **public static and final.**
- Also, few things are **going to be repeated** like **opening the browser, entering url and login credentials, verifying login page and closing the browser** etc. So, instead of writing the same in all the test scripts, we have created a **BaseClass** which was **extended** by all the test scripts without fail.
- Also, in **BaseClass**, we had included the **precondition** and **postcondition annotations** of **TestNG** for a **smooth execution.**
- To reduce the **complexity** of test scripts, and to avoid getting **exceptions** like **StaleElementReferenceException** etc, we used to create **POM classes** as an **object repository** wherein all the **elements** are **declared, initialized, and utilized** with the help of **public getters methods.,** and if **any modifications** are required, simply we used to **modify** the **POM classes** rather than changing them in all the test scripts.
- About **Test Data**, storing all the data inside the framework is a **tedious job**. Because it affects the **execution time** and since we can't go with **keyword driven** for **test data** as remembering all the keys is not possible, we used to **access the test data** from external resources like **excel file.**
- Sometimes, **depending upon** the **significance of data,** we used to implement **smart ways** like **faker class** and **random number classes** rather than **accessing** from **external files** all the time.
- About **common data**, I used to access it from **properties file** which include **url, login credentials etc.**
- Since everything was softcoded, the **length** and **dependency** between the **test cases** had been **reduced** by **utilizing** all the **generic libraries** inside each **test script** and hence, the **test scripts** became **light-weight.**
- Since, **TestNG** has a **beautiful feature** called **Batch Execution**, we used to **convert** all the **test classes** into **Test Suite** and we used to **run by a single click.**
- Since We were using **Maven** for creating the **Builds**, We used to look after the **integration** between **IDE, GitHub** and **Jenkins** accordingly**.**
- We used to utilize **IAutoListener** Interface for **screenshots,** especially for **failed scripts** by creating **Listener Implementation class** which contains all the **overridden methods.**
- We also included **retry()** of **IRetryAnalyzer** interface to **re-execute** the **failed test scripts**.
- I Used to **analyze the errors** in a test script with the help of **breakpoints** and **Expressions** window in **Debug mode.**
- Finally, I used to go with **ExtentReports** for all the emailable reports.

INCLUSION OF JAVA CONCEPTS IN MY FRAMEWORK:

- We included the **Inheritance** concept in **BaseClass** wherein all the **commonly repeated actions** are **stored** and ensured that **every test class extends** the **BaseClass**.
- Since the **elements** are **private** in each **POM class**, and is **accessible** through **public getters methods**. Here, we had utilized the concepts of both **Encapsulation** and **Abstraction**.
- Since the **reference variable** called **driver** can be **loaded** with either **FirefoxDriver object** or **ChromeDriver Object** in **BaseClass** based upon the browser value. The concept called **Polymorphism** was involved here.
- Since we **stored** all the **constants** which are the **same throughout the project** in an **Interface** called **IPathConstants**, The concept of **Interface** was also utilized here.
- We created a few **reusable methods** in the **Generic Library** out of which few are created based on **Method-overloading** concept.
- To take the **screenshots** of **failed test scripts**, we had **overridden** methods of **IAutoListener interface** which is being **implemented** by **ListenerClass**.So, the concept of **Method-Overridden** was also included here.
- Since the methods like **openBrowser()** and **closeBrowser()** of BaseClass are **dependent** on **@Test methods** for execution, we made **BaseClass** as an **Abstract class** so that testers can't run it. Here, the concept of **Abstract Class** was included.
- The concepts like **List** and **Set** interfaces of **Collection Framework** were utilized here to handle multiple elements and avoid duplicates respectively.
- In the **DataProvider** concept of **TestNG**, in order to store the data in data banks. We created an **object** of **two dimensional array** which is a Java concept.