Mobile Automation Framework for Grocery App with BrowserStack

1. Framework Type: Hybrid (Data-Driven + Page Object Model)

A hybrid automation framework ensures **reusability, scalability, and maintainability** by integrating **Page Object Model (POM)** for UI elements and **Data-Driven Testing (DDT)** for test execution with dynamic data.

2. Tools & Tech Stack

- **Programming Language:** Java
- **Automation Tool:** Appium (for Mobile Testing)
- **Test Framework:** TestNG
- **Build Management:** Maven
- **Reporting:** Extent Reports
- **Version Control:** GitHub
- **Cloud Execution:** BrowserStack

---

3. Framework Explanation

- My framework contains nine components they are,
  1.Utilities
  2.POM/ (object repository)
  3.TestData
  4.Resources
  5.Test Scripts
  6.Suite file
  7.POM.Xml
  8.AdvanceReport
  9..m2 folder

  1.**<u>Utilities:</u>**
  Utilities is the library/method which is common for all the projects.
  In this utility Package I have created File utility, Excel utility, Android Driver Utility, BrowserStackBaseclass, Object repository utility, listener utility classes for each class. I have created objects for all these classes in order to call the methods of these utilities.
  1.<u>FileUtility:</u>
  This is the class where I have created methods, which are required to get data from a property file.

2.<u>ExcelUtility:</u>
This is the class where I have created methods, which are required to get data from Excel files.

3.<u>AndroiddriverUtility:</u>

This is the class where I have created methods, which are required to do actions related to the Mobile application.
Example: Install app, uninstall app, hide keyboard etc.
4.<u>BrowserStackBaseclassUtility:</u>
This is the class where I have stored all the browserstack Configuration in the form of Desired capabilities and annotations like @BC,@AC,@BM,@AM I extended my test scripts to base class to do configurations for test annotation.

5.<u>ListenerUtility:</u>
This is the class where I have created a listener implementation class and this class implements ITestlistener, ISuitelistener these are interfaces so that we can call all the abstract methods which are present in both interfaces. I have provided implementations to abstract methods.
I have used @Listener in my test script in order to monitor my test script and this is responsible for capturing the runtime event, mainly we will use this to take the screenshot of the failed test scripts.

6.<u>GestureUtility:</u>
This utility will help to perform touch actions like scroll, tap, swipe etc.

7.<u>UtiliyObject:</u>
It will handle threads for each session execution and maintain the test session based on their thread instance id.

2**<u>.POM/Object Repository:</u>**
In this class I have stored all Mobile Elements which are present in the Mobile app page of an application. Pom class also contains business methods which are common actions in an application for example login(),logout() etc.

3.**<u>TestData:</u>**
In test data we have two types they are,
>Common data-Data which is common for all the test scripts,
>Test script data-Data which is specific for a particular test script.
I have created a folder in which I have stored common data by using property files(.properties) and test script data by using Excel file(.xlsx).

4.**<u>Test Scripts:</u>**
In the package I have created test scripts module wise.

During the development of the test script I have used all the generic utilities, POM(object repository), Testdata.

5.**<u>Resources:</u>**
Also we can store project related documents like Test cases, Excel files etc. Internally base class is connecting to this while launching the Application.

6.**<u>Suite file:</u>**
Once test scripts were developed I have converted all the files into TestNG file (testng.xml)/suite file.This suite file contains all the testNG classes.
When we run this suite file internally the suite file will trigger all the test scripts and generate the HTML report.

7.**<u>POM.Xml:</u>**
During the development of the framework I have used maven project so I got by default pom.xml, this is a project configuration file where we are going to keep all dependencies which are required to get the libraries and predefined classes and methods of third party tools.

In my project I have added some of dependencies like,
>Appium Java client for device interaction
>selenium webdriver device interaction
>Apache poi to excel interaction
>testNG to execute suite
>Extentreports for test execution

8.**<u>AdvanceReport:</u>**
Once the test execution completes default test reports stored in the form of html which contains timestamp and testmethod name.

9. **<u>M2 Folder:</u>**
It used to store all the dependency jar files of the project by default.

Conclusion:
This framework ensures **scalability, maintainability, and efficient automation testing** for a **Grocery Store mobile app** using **Appium, TestNG, and BrowserStack**. It allows running tests on **multiple real devices in the cloud**, reducing execution time and improving test coverage.