What is Framework?

A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be incorporated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

Let us consider a real-life scenario.
We very often use lifts or elevators. There are a few guidelines those are mentioned within the elevator to be followed and taken care off so as to leverage the maximum benefit and prolonged service from the system.

Thus, the users might have noticed the following guidelines:

- Keep a check on the maximum capacity of the elevator and do not get onto an elevator if the maximum capacity has reached.
- Press the alarm button in case of any emergency or trouble.
- Allow the passenger to get off the elevator if any before entering the elevator and stand clear of the doors.
- In case of fire in the building or if there is any haphazard situation, avoid the use of the elevator.
- Do not play or jump inside the elevator.
- Do not smoke inside the elevator.
- Call for the help/assistance if the door doesn't open or if the elevator doesn't work at all. Do not try to open the doors forcefully.

There can be many more rules or sets of guidelines. Thus, these guidelines if followed makes the system more beneficial, accessible, scalable and less troubled for the users.

Now, as we are talking about "Test Automation Frameworks", let us move our focus towards them.

Test Automation Framework

A "Test Automation Framework" is scaffolding that is laid to provide an execution environment for the automation test scripts. The framework provides the user with various benefits that help them to develop, execute and report the automation test scripts efficiently. It is more like a system that has created specifically to automate our tests.

In a very simple language, we can say that a framework is a constructive blend of various guidelines, coding standards, concepts, processes, practices, project hierarchies, modularity, reporting mechanism, test data injections etc. to pillar automation testing. Thus, the user can follow these guidelines while automating application to take advantages of various productive results.

The advantages can be in different forms like the ease of scripting, scalability, modularity, understandability, process definition, re-usability, cost, maintenance etc. Thus, to be able to grab these benefits, developers are advised to use one or more of the Test Automation Framework.

Moreover, the need of a single and standard Test Automation Framework arises when you have a bunch of developers working on the different modules of the same application and when we want to avoid situations where each of the developers implements his/her approach towards automation.

Note: Take a note that a testing framework is always application independent that is it can be used with any application irrespective of the complications (like Technology stack, architecture etc.) of the application under test. **The framework should be scalable and maintainable.**
Advantage of Test Automation framework

1. Reusability of code
2. Maximum coverage
3. Recovery scenario
4. Low-cost maintenance
5. Minimal manual intervention
6. Easy Reporting

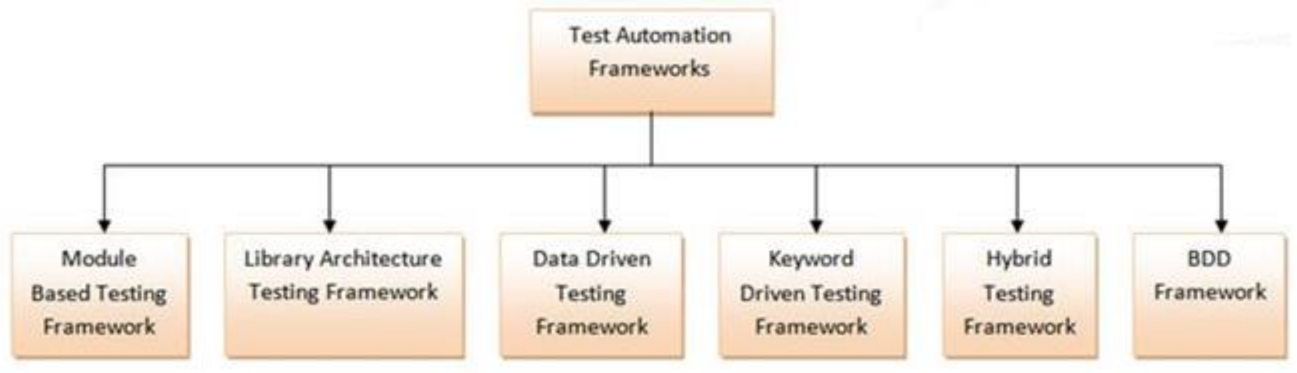
Types of Test Automation Framework

Now that we have a basic idea of what is an Automation Framework, in this section we would harbinger you with the various types of Test Automation Frameworks those are available in the marketplace. We would also try shed lights over their pros and cons and usability recommendations.

There is a divergent range of Automation Frameworks available nowadays. These frameworks may differ from each other based on their support to different key factors to do automation like reusability, ease of maintenance etc.

Let us discuss the few most popularly used Test Automation Frameworks:

1. Module Based Testing Framework
2. Library Architecture Testing Framework
3. Data Driven Testing Framework
4. Keyword Driven Testing Framework
5. Hybrid Testing Framework
6. Behavior Driven Development Framework


Test Automation Frameworks
Module Based Testing Framework
Library Architecture Testing Framework
Data Driven Testing Framework
Keyword Driven Testing Framework
Hybrid Testing Framework
BDD Framework

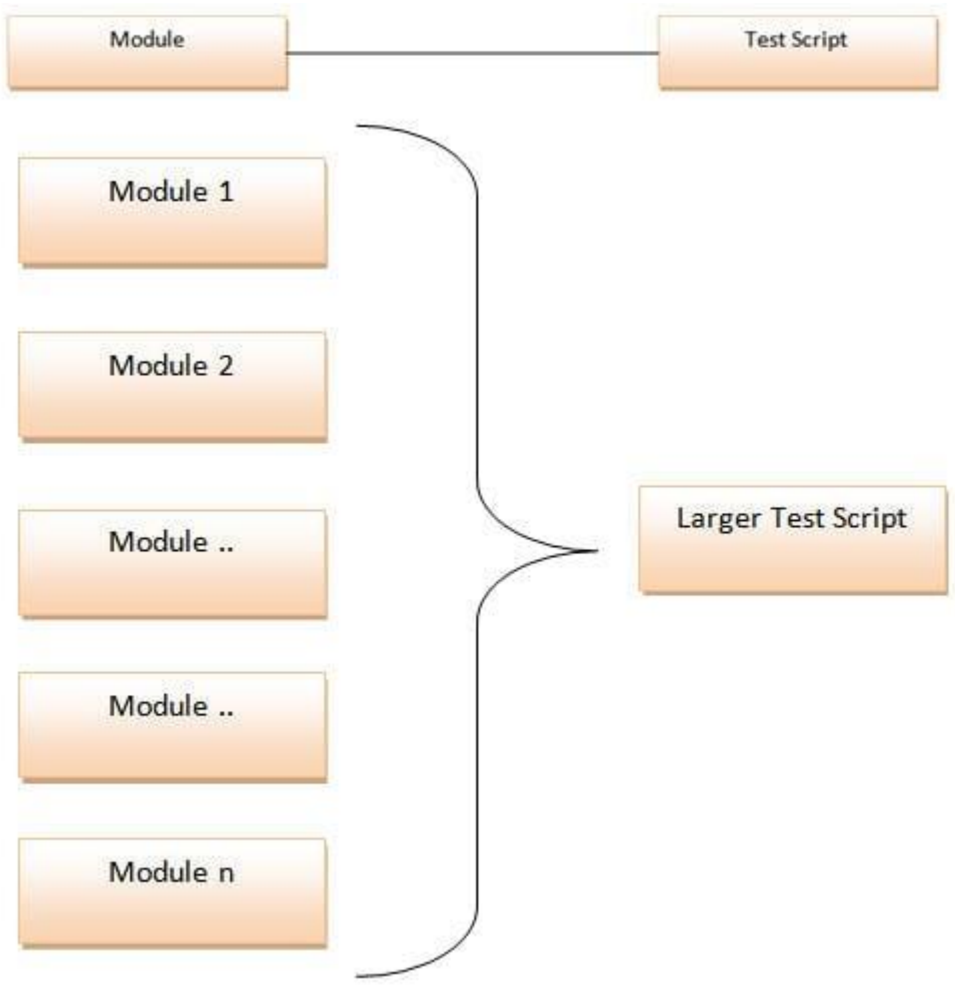
Let us discuss each of them in detail.

But before that, I would also like to mention that despite having this framework, the user is always leveraged to build and design his own framework which is best suitable to his/her project needs.

#1) Module Based Testing Framework

Module based Testing Framework is based on one of the popularly known OOPs concept – Abstraction. The framework divides the entire "Application Under Test" into a number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts took together builds a larger test script representing more than one modules.

These modules are separated by an abstraction layer in such a way that the changes made in the sections of the application doesn't yield affects on this module.



Pros:

1. The framework introduces the high level of modularization which leads to easier and cost-efficient maintenance.
2. The framework is pretty much scalable
3. If the changes are implemented in one part of the application, only the test script representing that part of the application needs to be fixed to leave all the other parts untouched.

Cons:

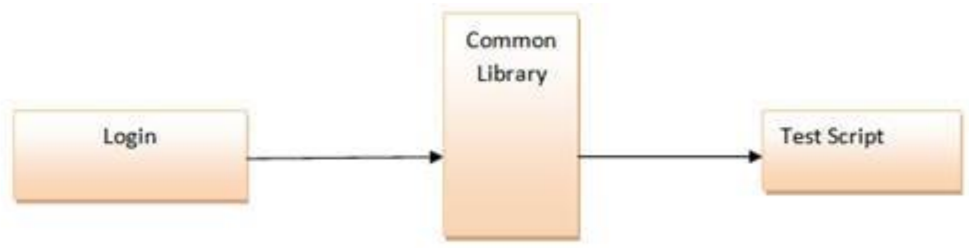
1. While implementing test scripts for each module separately, we embed the test data (Data with which we are supposed to perform testing) into the test scripts. Thus, whenever we are supposed to test with a different set of test data, it requires the manipulations to be made in the test scripts.

#2) Library Architecture Testing Framework

The Library Architecture Testing Framework is fundamentally and foundationally built on Module Based Testing Framework with some additional advantages. Instead of dividing the application under test into test scripts, we segregate the application into functions or rather common functions can be used by the other parts of the application as well. Thus we create a common library constituting of common functions for the application under test. Therefore, these libraries can be called from the test scripts whenever required.

The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

<u>**Example**</u>: The login steps can be combined into a function and kept into a library. Thus all the test scripts those require to login the application can call that function instead of writing the code all over again.



Pros:
1. Like Module Based Framework, this framework also introduces the high level of modularization which leads to easier and cost-efficient maintenance and scalability too.
2. As we create common functions that can be efficiently used by the various test scripts across the Framework. Thus, the framework introduces a great degree of re-usability.

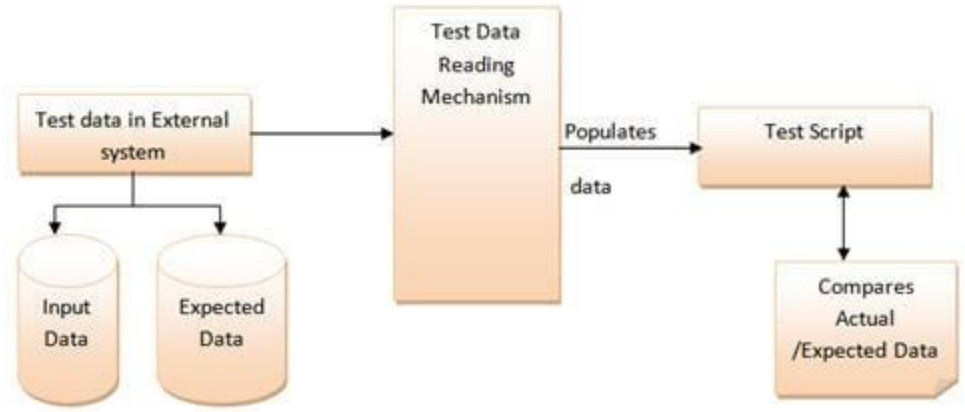
Cons:
1. Like Module Based Framework, the test data is lodged into the test scripts, thus any change in the test data would require changes in the test script as well.
2. With the introduction of libraries, the framework becomes a little complicated.

#3) Data Driven Testing Framework

While automating or testing any application, at times it may be required to test the same functionality multiple times with the different set of input data. Thus, in such cases, we can’t let the test data embedded in the test script. Hence it is advised to retain test data into some external database outside the test scripts.

Data Driven Testing Framework helps the user segregate the test script logic and the test data from each other. It lets the user store the test data into an external database. The external databases can be property files, xml files, excel files, text files, CSV files, ODBC repositories etc. The data is conventionally stored in “Key-Value” pairs. Thus, the key can be used to access and populate the data within the test scripts.

Note: The test data stored in an external file can belong to the matrix of expected value as well as the matrix of input values.

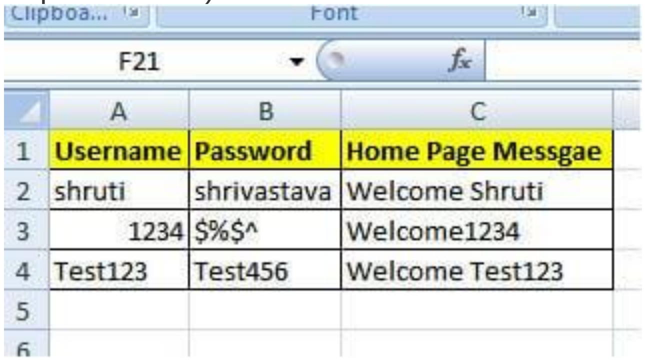

Test Data Reading Mechanism
Test data in External system
Populates data
Test Script
Input Data
Expected Data
Compares Actual /Expected Data

Example:

Let us understand the above mechanism with the help of an example.

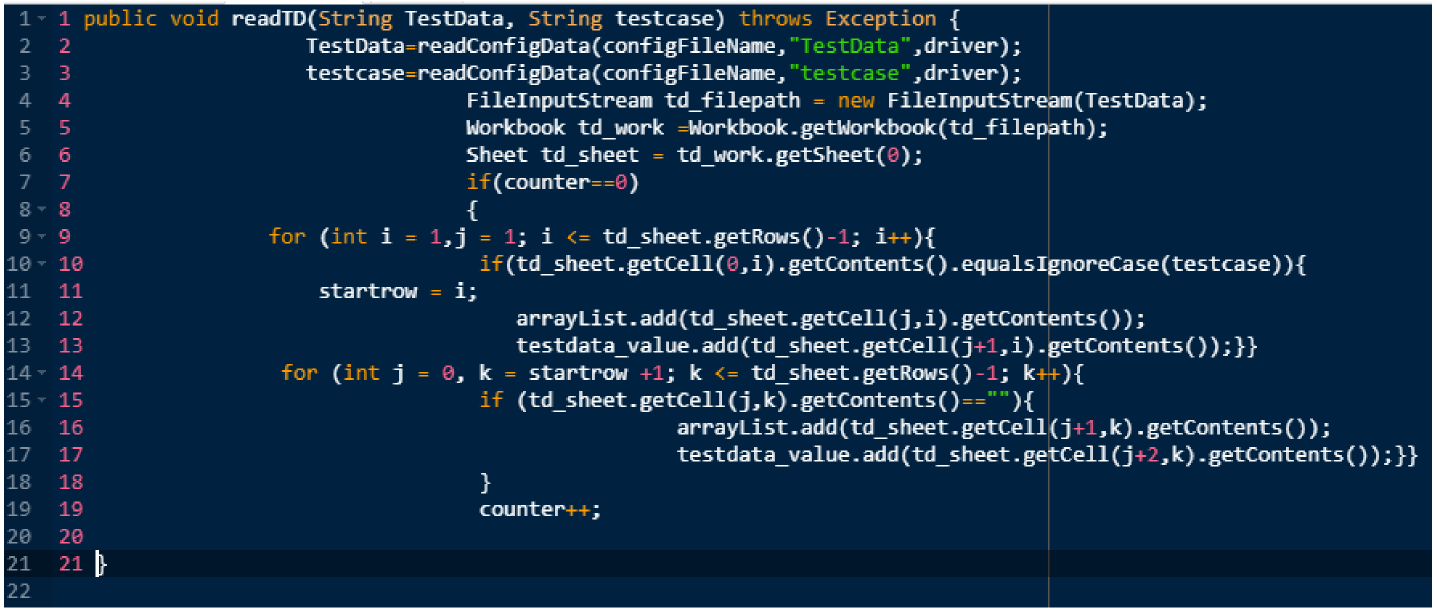
Let us consider the “Gmail – Login” Functionality.

Step 1: First and the foremost step are to create an external file that stores the test data (Input data and Expected Data). Let us consider an excel sheet for instance.



	A	B	C
1	Username	Password	Home Page Messgae
2	shruti	shrivastava	Welcome Shruti
3	1234	$%$^	Welcome1234
4	Test123	Test456	Welcome Test123
5			
6			

Step 2: The next step is to populate the test data into Automation test Script. For this purpose, several API’s can be used to read the test data.



```
public void readTD(String TestData, String testcase) throws Exception {
                  TestData=readConfigData(configFileName,"TestData",driver);
                  testcase=readConfigData(configFileName,"testcase",driver);
                               FileInputStream td_filepath = new FileInputStream(TestData);
                               Workbook td_work =Workbook.getWorkbook(td_filepath);
                               Sheet td_sheet = td_work.getSheet(0);
                               if(counter==0)
                               {
               for (int i = 1,j = 1; i <= td_sheet.getRows()-1; i++){
                                if(td_sheet.getCell(0,i).getContents().equalsIgnoreCase(testcase)){
                   startrow = i;
                                   arrayList.add(td_sheet.getCell(j,i).getContents());
                                   testdata_value.add(td_sheet.getCell(j+1,i).getContents());}}
                for (int j = 0, k = startrow +1; k <= td_sheet.getRows()-1; k++){
                                if (td_sheet.getCell(j,k).getContents()==""){
                                             arrayList.add(td_sheet.getCell(j+1,k).getContents());
                                             testdata_value.add(td_sheet.getCell(j+2,k).getContents());}}
                                }
                                counter++;

}
```

The above method helps to read the test data and the below test step helps the user to type in the test data on the GUI.

element.sendKeys(obj_value.get(obj_index));

Pros:

1. The most important feature of this framework is that it considerably reduces the total number of scripts required to cover all the possible combinations of test scenarios. Thus lesser amount of code is required to test a complete set of scenarios.
2. Any change in the test data matrix would not hamper the test script code.
3. Increases flexibility and maintainability
4. A single test scenario can be executed altering the test data values.

Cons:

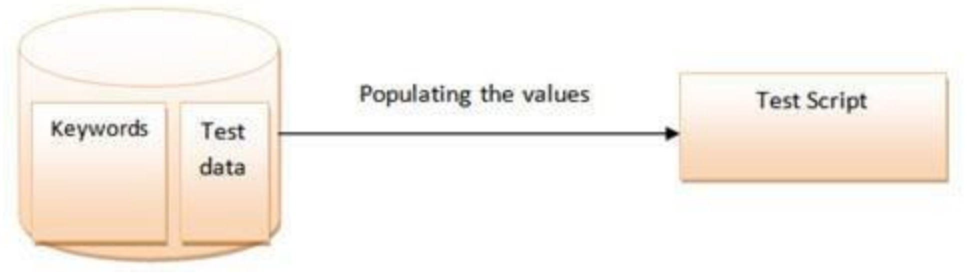
1. The process is complex and requires an extra effort to come up with the test data sources and reading mechanisms.
2. Requires proficiency in a programming language that is being used to develop test scripts.

#4) Keyword Driven Testing Framework

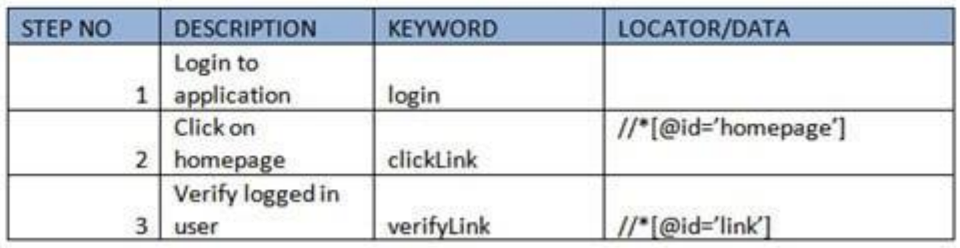
The Keyword driven testing framework is an extension to Data driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.

These set of code are known as Keywords and hence the framework is so named. Keywords are self-guiding as to what actions need to be performed on the application.

The keywords and the test data are stored in a tabular like structure and thus it is also popularly regarded as Table driven Framework. Take a notice that keywords and test data are entities independent of the automation tool being used.



<u>Example</u> Test case of Keyword Driven Test Framework



STEP NO	DESCRIPTION	KEYWORD	LOCATOR/DATA
1	Login to application	login	
2	Click on homepage	clickLink	//*[@id='homepage']
3	Verify logged in user	verifyLink	//*[@id='link']

In the above example, keywords like login, clicking and verify Link are defined within the code. Depending upon the nature of application keywords can be derived. And all the keywords can be reused multiple times in a single test case. Locator column contains the locator value that is used to identify the web elements on the screen or the test data that needs to be supplied.

All the required keywords are designed and placed in the base code of the framework.

Pros:

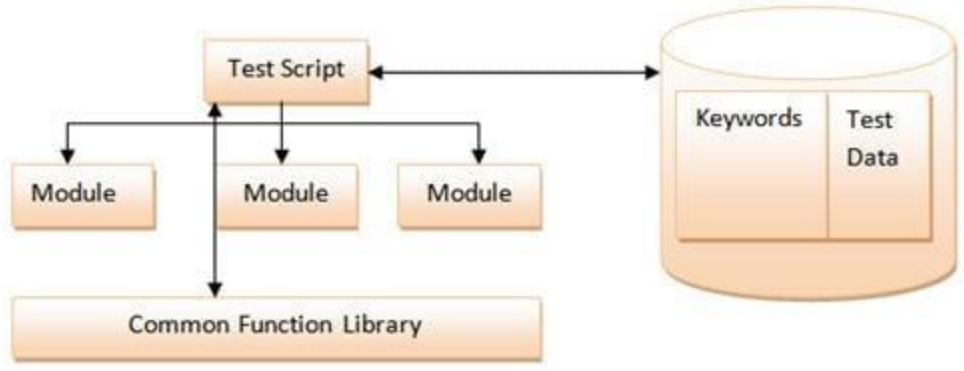
1. In addition to advantages provided by Data Driven testing, the Keyword driven framework doesn't require the user to possess scripting knowledge, unlike Data Driven Testing.
2. A single keyword can be used across multiple test scripts.

Cons:

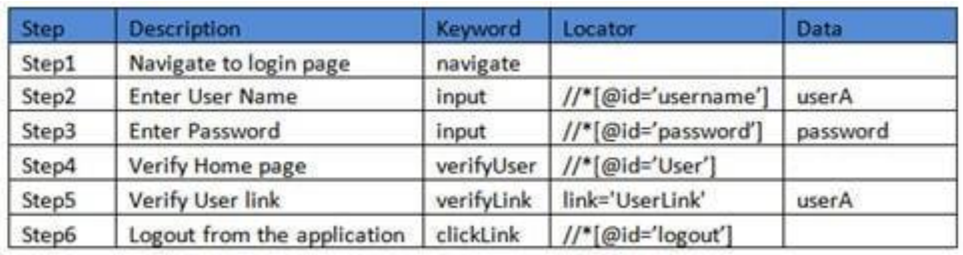
1. The user should be well versed with the Keyword creation mechanism to be able to efficiently leverage the benefits provided by the framework.
2. The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

#5) Hybrid Testing Framework

As the name suggests, the Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.



<u>Example</u> of Hybrid Framework
Test sheet would contain both the keywords and the Data.



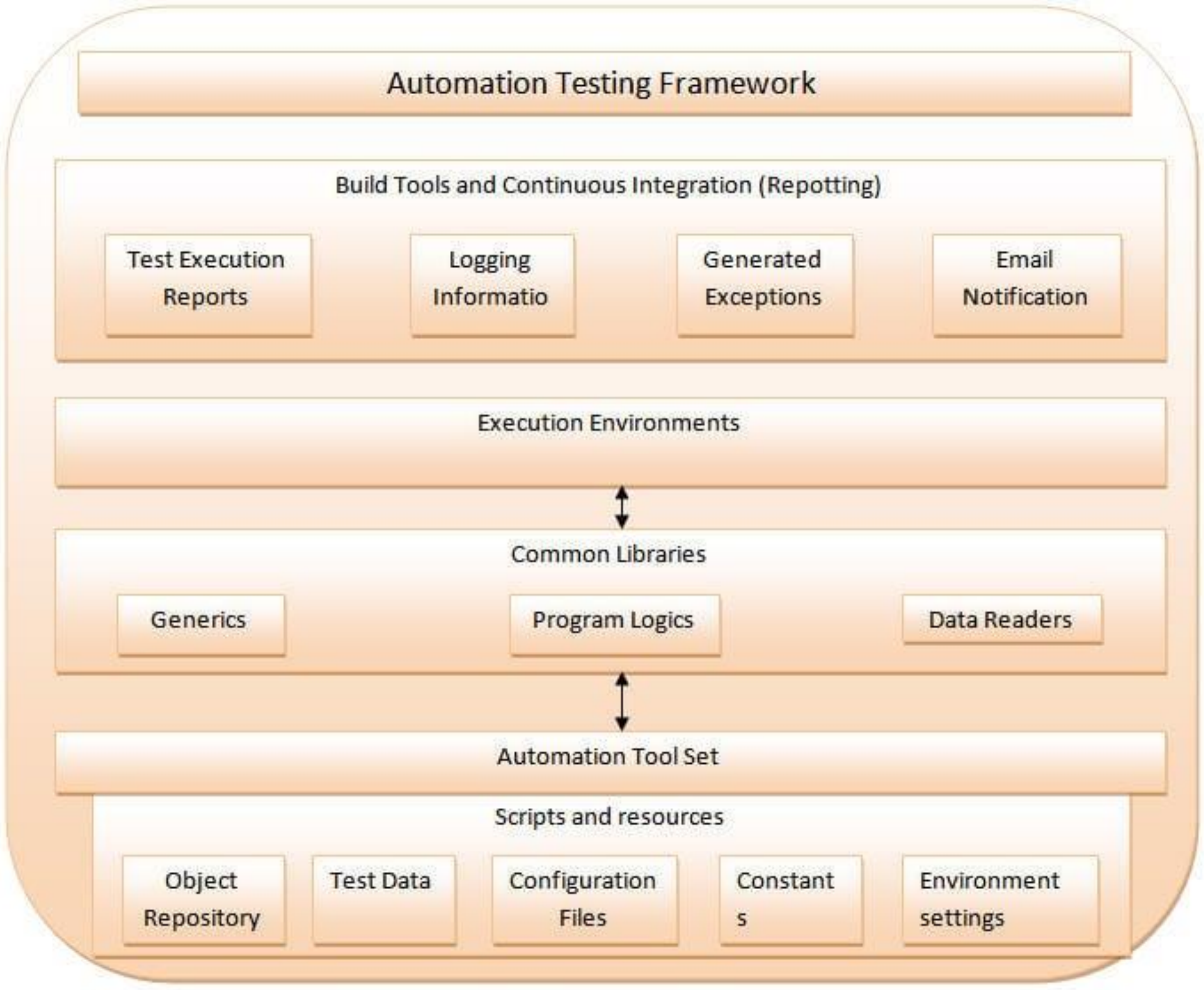
Step	Description	Keyword	Locator	Data
Step1	Navigate to login page	navigate		
Step2	Enter User Name	input	//*[@id='username']	userA
Step3	Enter Password	input	//*[@id='password']	password
Step4	Verify Home page	verifyUser	//*[@id='User']	
Step5	Verify User link	verifyLink	link='UserLink'	userA
Step6	Logout from the application	clickLink	//*[@id='logout']	

In the above example, keyword column contains all the required keywords used in the particular test case and data column drives all the data required in the test scenario. If any step does not need any input then it can be left empty.

#6) Behavior Driven Development Framework

Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc. Such frameworks do not necessarily require the user to be acquainted with the programming language. There are different tools available for BDD like cucumber, Jbehave etc. Details of BDD framework are discussed later in Cucumber tutorial. We have also discussed details on Gherkin language to write test cases in Cucumber.

<u>Components of Automation Testing Framework</u>
(click on image to view enlarged)



Though the above pictorial representation of a framework is self-explanatory we would still highlight a few points.

1. **Object Repository**: Object Repository acronym as OR is constituted of the set of locators types associated with web elements.
2. **Test Data:** The input data with which the scenario would be tested and it can be the expected values with which the actual results would be compared.
3. **Configuration File/Constants/ Environment Settings**: The file stores the information regarding the application URL, browser-specific information etc. It is generally the information that remains static throughout the framework.
4. **Generics/ Program logics/ Readers**: These are the classes that store the functions which can be commonly used across the entire framework.
5. **Build tools and Continuous Integration**: These are the tools that aids to the capabilities of the framework to generate test reports, email notifications and logging information.

Conclusion

The frameworks illustrated above are the most popular frameworks used by the testing fraternity. There are various other frameworks also in the place. For all the further tutorials we would base on the **Data Driven Testing Framework**.

In this tutorial, we discussed the basics of an Automation Framework. We also discussed the types of frameworks available in the market.