1. **Technologies Used**
   1. Java 1.8
   2. Maven
   3. Selenium 3
   4. Test NG 7.4
   5. Log 4j 4.1
   6. Extent Report 2.41.2
   7. RestAssured 4.1.1
   8. Page Object Model(POM) Design Pattern
2. **Features** 
   1. Logger – Log4j has been used. Created Log.java file which has the following methods
      1. startLogger()
      2. endLogger()

These two methods are called in each before and after each test case. Log.xml file is generated which can be referred to check the logs

**Log.xml path - \picsartsite\log.xml**

* 1. Extent Report – Extent Report has been used for generating the test report. As for now showing Test case status can be extend further to capture complete steps details

**Extent Report path - \picsartsite\PicsArtReport.html**

* 1. Parallel Execution using Test NG – Test cases runs in parallel to save the execution time.

In this framework **parallel execution** is done by using the **TestNG**.

* 1. Test Case fail try mechanism
  2. Smart wait mechanism to wait for an element –

Before performing any action on an element smart wait has been implemented.

A wrapper class has been added with generic methods which first check for the visibility/availability of the element using the **explicit wait** before performing any action.

For example: for clicking any element with locator as xpath below method is called which first do the explicit wait to check whether the element is available or not . Once element is available only then it will click the element otherwise through the ‘Element Not Found exception’

**public** **static** **void** clickElementByXpath(WebElement ele,WebDriver driver) **throws** Exception

{

**try**

{

Log.*info*("Searching for element"+ele );

*wait* = **new** WebDriverWait(driver,*mediumWait*);

*wait*.until(ExpectedConditions.*elementToBeClickable*(ele));

Log.*info*(ele+" is clickable");

ele.click();

Thread.*sleep*(2000);

Log.*info*("Element: '"+ele +"' clicked successful.");

}

**catch**(StaleElementReferenceException stale)

{

Log.*info*("StaleElementReferenceException");

driver.navigate().refresh();

ele.click();

}

**catch** (NoSuchElementException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

Log.*info*("xpath : "+ele +"click by xpath failed");

**throw** **new** Exception("NoSuchElementException!");

}

**catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

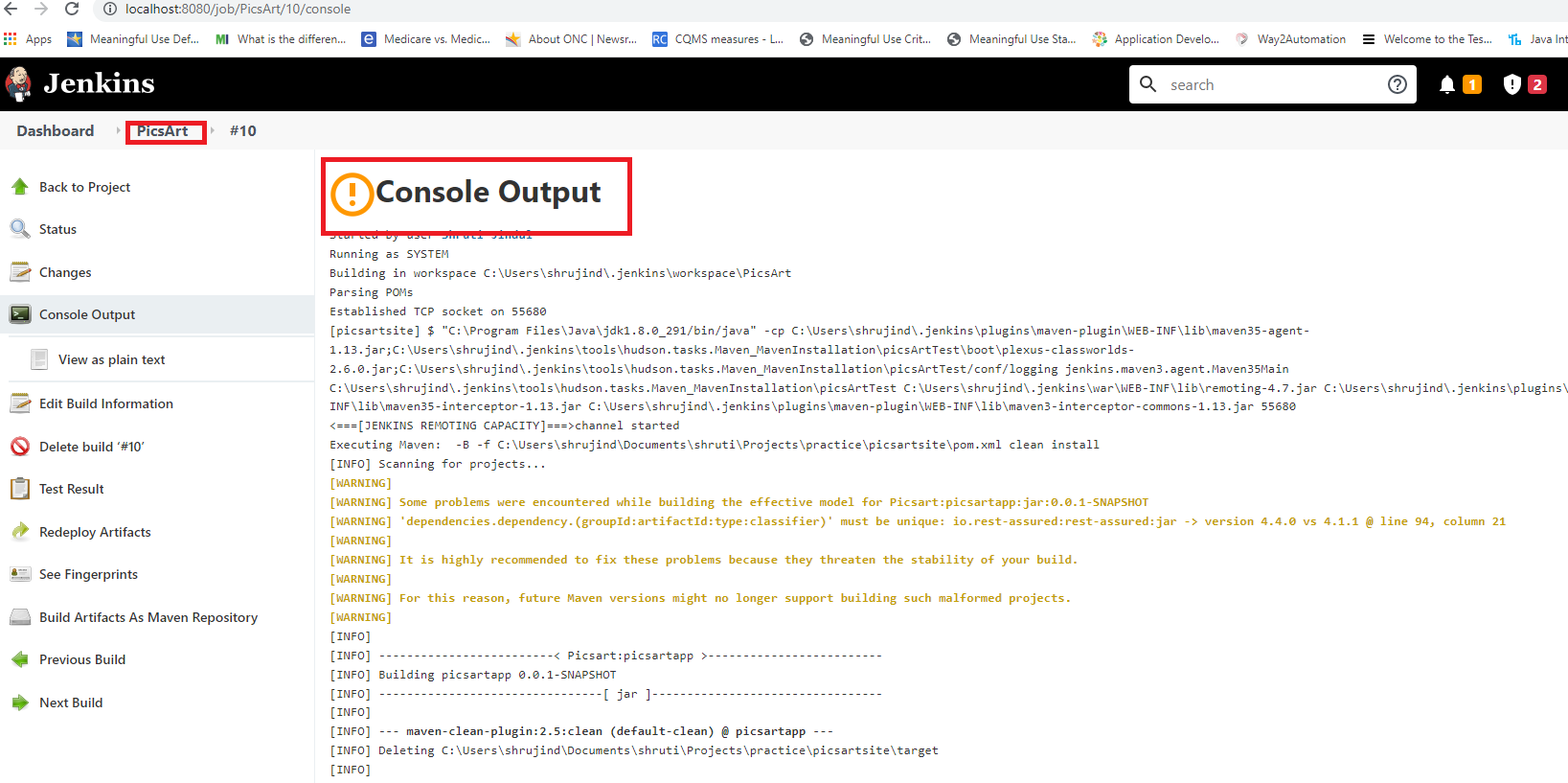
Log.*info*("xpath : "+ele +"click by xpath failed");

**throw** **new** Exception("Error at UI Execution!");

}

}

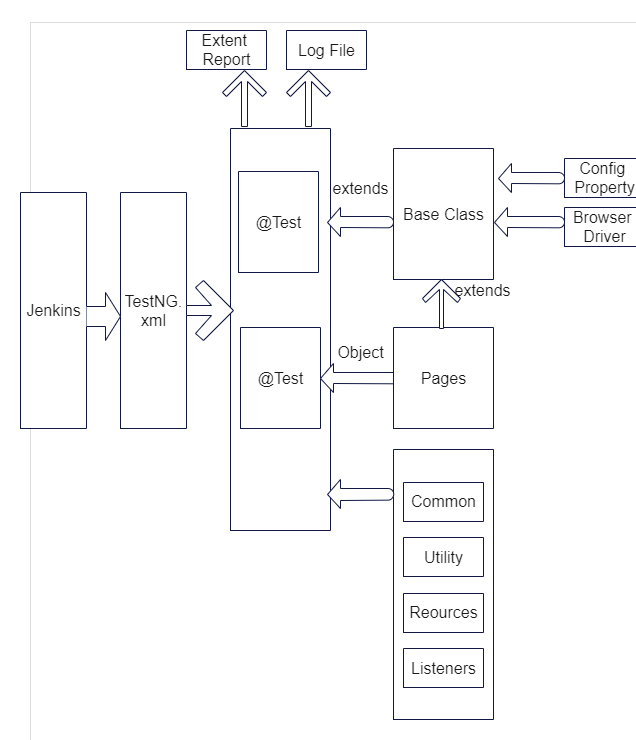
* 1. Integrated with Jenkins : Code has been integrated with the Jenkins. Build is running successfully.



* 1. Exception Handling : Proper Exception handling has been done for all the methods.

StaleReferenceException- In PicsArt site while navigating from 1 link to other Stale Reference exception was occurring . That has been handled using **driver.navigate.refresh()**

1. **Framework Flow**



1. **Framework Structure:**

Components:

1. Base – Consist Of Base class which loads the Config Property File and initiate the browser

All the page and test classes extends the Base Class

1. Page: Page package is consist page classes. As per POM design pattern each screen has one page class which consist of all the elements of the page and also initialize the webelements using the PageInit.Factory() method. All page classes extends the Base class.
2. Test: Test Folder is consist of all test classes. TestNG annotations are used. All test classes extends the Base class and create the object the Page classes to call the actions written in the page classes
3. Common: This folder is consist of common methods
   1. WebElementWrapper.java
   2. JsonschemaValidator.java -
4. Drivers : Consist of the browser drivers
5. Listeners : Consist of following listener classes
   1. ExtentReportListener : Listener class for generating the extent report
   2. RetryListenerClass – To retry the failed test cases
6. Properties: Consist of Config.properties . In this framework following values are read from config.properties :
   1. App url
   2. Username/Password
   3. Expected values
   4. DataInput
7. Resources: consist of
   1. Log4.xml – configuration file for generating the logs
   2. RetryFailedTestCases – This class is called by each test case after the execution. If the test case will be re-executed. Maximum number of try can be given here
8. Utilities : Utility classes
9. JsonSchema: is consist of the expected schema for the given api response. This schema can be used to validate the actual schema response recived

