How to Generate Extent Reports in Selenium

What are Extent Reports?

Using Extent Reports in Selenium

How to generate Extent Reports

How to capture screenshots in Extent Report

Benefits of using Extent Reports

### **What are Extent Reports?**

Extent Reports is an open-source reporting library useful for test automation. It can be easily integrated with major testing frameworks like JUnit, NUnit, TestNG, etc. These reports are HTML documents that depict results as pie charts. They also allow the generation of custom logs, snapshots, and other customized details.

Once an automated test script runs successfully, testers need to generate a test execution report. While TestNG does provide a default report, they do not provide the details.

ExtentReport API makes our life easy to generate interactive report with simple configuartions. It supports almost all Java and .NET test frameworks such as TestNG, JUnit, NUnit etc.

Extent Report Maven Dependency:

<**dependency**>

     <**groupId**>com.aventstack</**groupId**>

     <**artifactId**>extentreports</**artifactId**>

     <**version**>3.1.5</**version**>

</**dependency**>

**package** com.Reporting.Extent;

**import** java.util.Arrays;

**import** org.testng.Assert;

**import** org.testng.ITestResult;

**import** org.testng.SkipException;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**import** com.aventstack.extentreports.Status;

**import** com.aventstack.extentreports.markuputils.ExtentColor;

**import** com.aventstack.extentreports.markuputils.MarkupHelper;

**import** com.aventstack.extentreports.reporter.ExtentHtmlReporter;

**import** com.aventstack.extentreports.reporter.configuration.ChartLocation;

**import** com.aventstack.extentreports.reporter.configuration.Theme;

**public** **class** BasicExtentReport {

//builds a new report using the html template

ExtentHtmlReporter htmlReporter;

ExtentReports extent;

//helps to generate the logs in test report.

ExtentTest test;

@Parameters({ "OS", "browser" })

@BeforeTest

**public** **void** startReport(String OS, String browser) {

// initialize the HtmlReporter

htmlReporter = **new** ExtentHtmlReporter(System.*getProperty*("user.dir") +"/test-output/testReport.html");

//initialize ExtentReports and attach the HtmlReporter

extent = **new** ExtentReports();

extent.attachReporter(htmlReporter);

//To add system or environment info by using the setSystemInfo method.

extent.setSystemInfo("OS", OS);

extent.setSystemInfo("Browser", browser);

//configuration items to change the look and feel

//add content, manage tests etc

htmlReporter.config().setChartVisibilityOnOpen(**true**);

htmlReporter.config().setDocumentTitle("Extent Report Demo");

htmlReporter.config().setReportName("Test Report");

htmlReporter.config().setTestViewChartLocation(ChartLocation.***TOP***);

htmlReporter.config().setTheme(Theme.***STANDARD***);

}

@Test

**public** **void** testCase1() {

test = extent.createTest("Test Case 1", "PASSED test case");

Assert.*assertTrue*(**true**);

test.log(Status.***INFO***,"Test Case 2");

}

@Test

**public** **void** testCase2() {

test = extent.createTest("Test Case 2", "PASSED test case");

Assert.*assertTrue*(**true**);

test.log(Status.***INFO***,"Test Case 2");

}

@Test

**public** **void** testCase3() {

test = extent.createTest("Test Case 3", "PASSED test case");

Assert.*assertTrue*(**true**);

}

@Test

**public** **void** testCase4() {

test = extent.createTest("Test Case 4", "PASSED test case");

Assert.*assertTrue*(**false**);

}

@Test

**public** **void** testCase5() {

test = extent.createTest("Test Case 5", "SKIPPED test case");

**throw** **new** SkipException("Skipping this test with exception");

}

@Test(enabled=**false**)

**public** **void** testCase6(){

test = extent.createTest("Test Case 6", "I'm Not Ready, please don't execute me");

}

@AfterMethod

**public** **void** getResult(ITestResult result) {

**if**(result.getStatus() == ITestResult.***FAILURE***) {

test.log(Status.***FAIL***, MarkupHelper.*createLabel*(result.getName()+" FAILED ", ExtentColor.***RED***));

//test.fail(result.getThrowable());

String excepionMessage = Arrays.*toString*(result.getThrowable().getStackTrace());

test.fail("<b>" + "<font color=" + "red>" + "Exception Occured:Click to see"

+ "</font>" + "</b >" + excepionMessage.replaceAll(",", "<br>")

+ " \n");

}

**else** **if**(result.getStatus() == ITestResult.***SUCCESS***) {

test.log(Status.***PASS***, MarkupHelper.*createLabel*(result.getName()+" PASSED ", ExtentColor.***GREEN***));

}

**else** {

test.log(Status.***SKIP***, MarkupHelper.*createLabel*(result.getName()+" SKIPPED ", ExtentColor.***ORANGE***));

test.skip(result.getThrowable());

}

}

@AfterTest

**public** **void** tearDown() {

//to write or update test information to reporter

extent.flush();

}

### }

'ExtentHtmlReporter' creates a rich standalone HTML file. It accepts ExtentHtmlReporter(java.io.File file) and ExtentHtmlReporter(java.lang.String filePath). In the above code, we are passing file path where the reports will be saved.

'ExtentHtmlReporter' also allows several configuration options via the config() method. In the above code, we have added few configuration items to change the look and feel of the report like report name, time stamp etc.

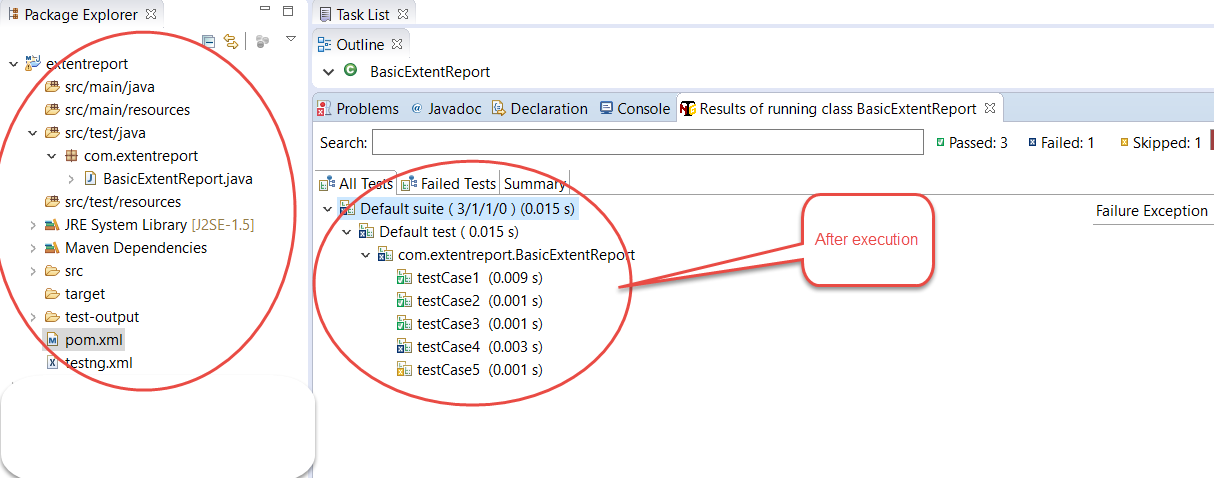
In 'AfterMethod', we are getting the status of each test executed and updating the status with **Fail**, **Pass** and **Skip**. If the test is failed, we are getting message string that gives more information about the errorresult.getThrowable()

**MarkupHelper -**MarkupHelper class is used to create labels like 'PASSED' / 'FAILED' and also to give colors to the different status in the test report using ExtentColor enum.

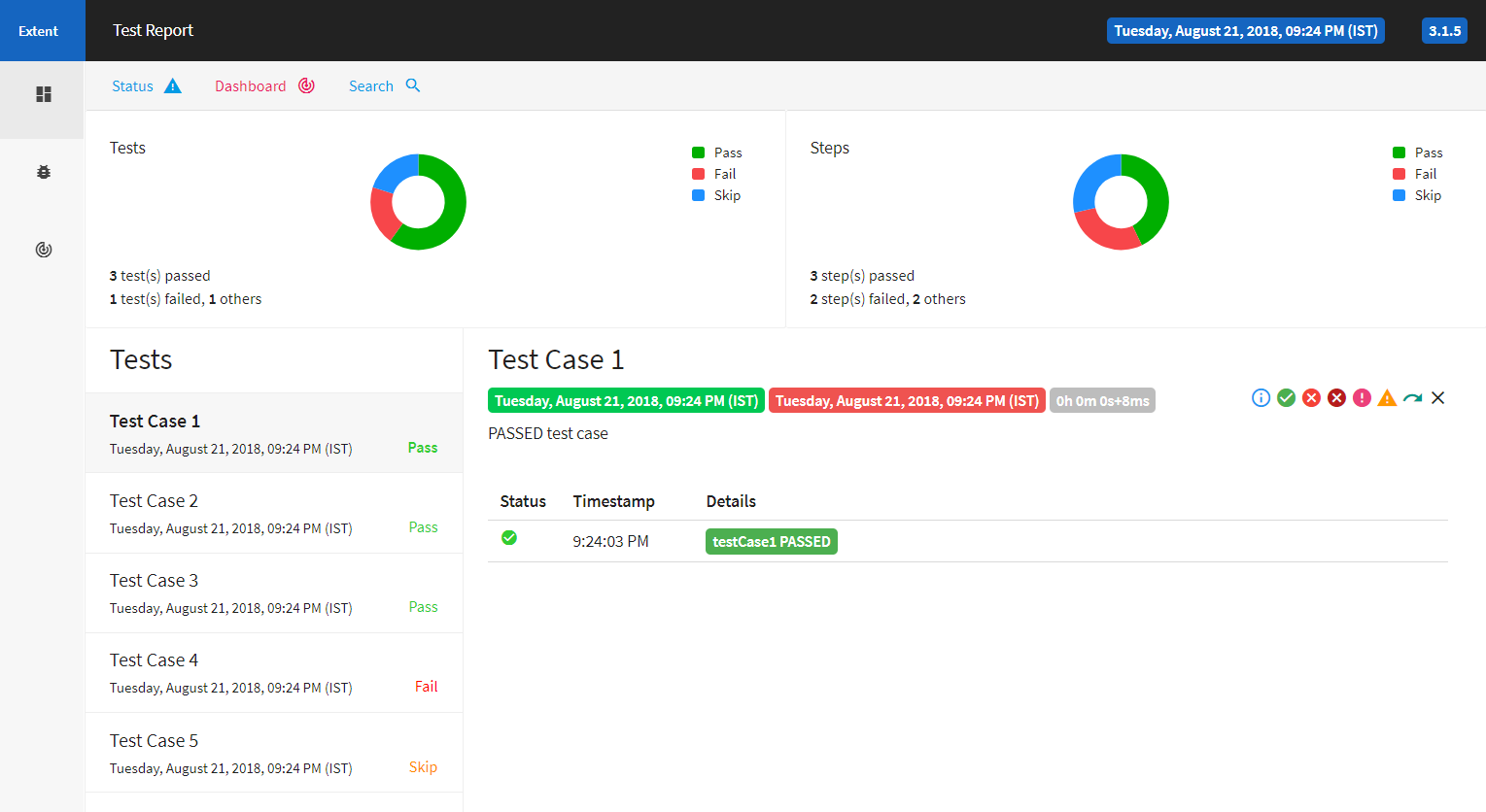
**Step 2 :-**Create testng.xml file like below to pass OS and Browser parameters : -

<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE suite SYSTEM "[http://testng.org/testng-1.0.dtd"](http://testng.org/testng-1.0.dtd) > <**suite** name="Main Test Suite" verbose="5"> <**parameter** name="OS" value="Windows"/> <**parameter** name="browser" value="Chrome"/>   <**test** name="Extent Report Test">    <**classes**>      <**class** name="com.extentreport.BasicExtentReport"/>     </**classes**>   </**test**> </**suite**>

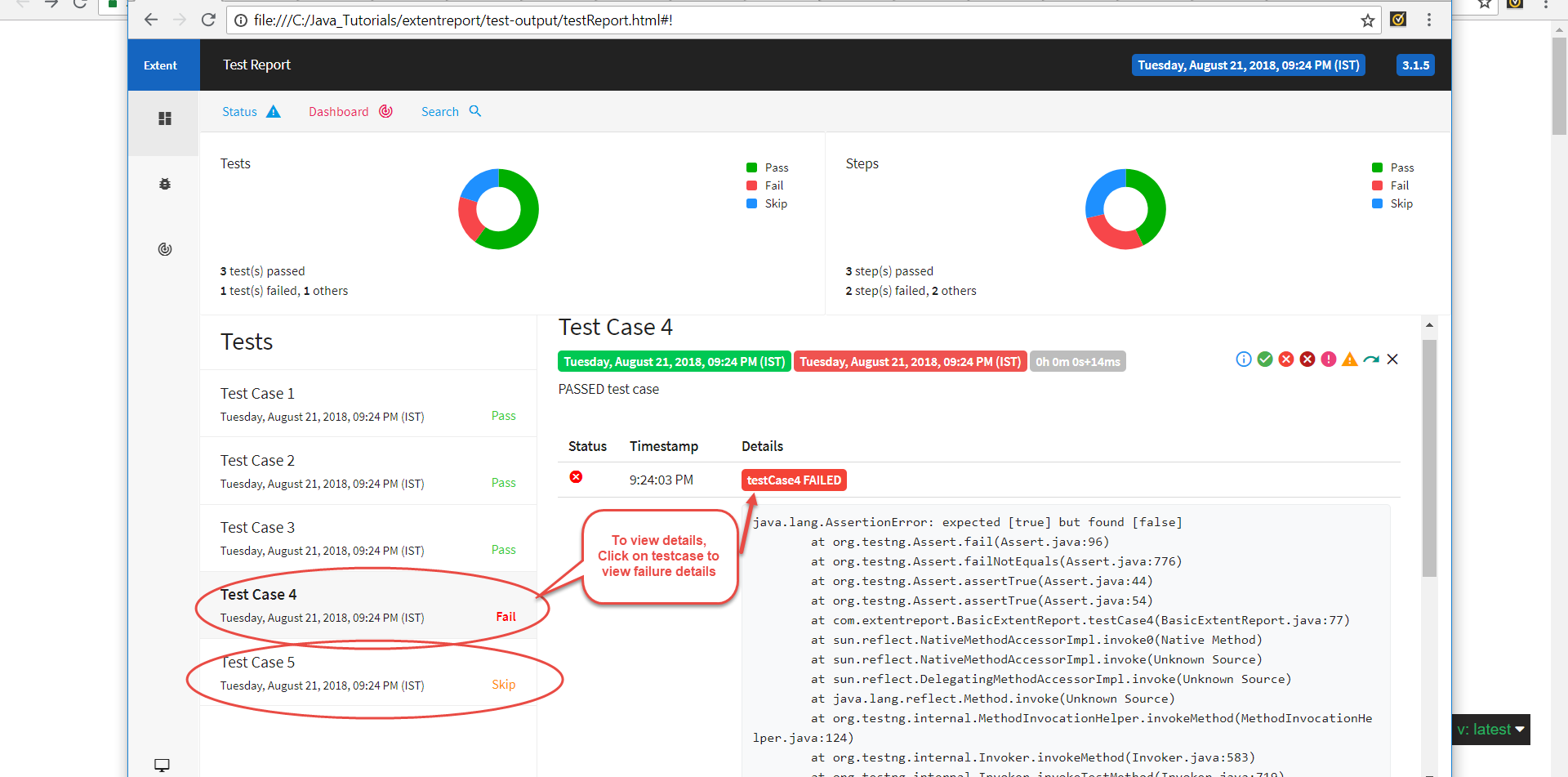
**Once we run the program, you should the report like below : -**  
**Below is the output image :-**



**Below is the Extent Report HTML looks like : -**



If you want to view the failure reason, you just need to click on failed test case which will show the status of the test and the reason of the failure (exception details). Below is the failed test case view.



**Extent Reports using TestNG Listeners in POM**

As we know all know that ExtentReport is an HTML reporting library (open source) which can be integrated with Selenium WebDriver.

**Importance of Reporting in a Test Automation Framework** :-

Even if you have the best automation framework, without reporting it's difficult to know just what went wrong and where should we fix if it is an automation issue. These reports help us to make important decisions as per the release requirement. Automated reporting provide valuable insights into not only the big picture of the tests executed, also the details of each test case passed and failed with screenshots

**Page Object Model** is a design pattern which is very popular in test automation for enhancing *test maintenance* and reducing *code duplication*. POM is best for the applications which has multiple pages.

If your not familiar using page object model, please refer to understand how page object design patterns works. We will be using page object model example from previous article where we already created page objects and tests. And will add few changes to integrate extent reports using TestNG ITestListeners.

### **Let’s look at page object model example and add ExtentReports : -**

1. Create a maven project and add below dependencies to pom.xml

*<!--* [*https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-server*](https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-server) *-->* <**dependency**>     <**groupId**>org.seleniumhq.selenium</**groupId**>     <**artifactId**>selenium-server</**artifactId**>     <**version**>3.14.0</**version**> </**dependency**> *<!--* [*https://mvnrepository.com/artifact/org.testng/testng*](https://mvnrepository.com/artifact/org.testng/testng) *-->* <**dependency**>     <**groupId**>org.testng</**groupId**>     <**artifactId**>testng</**artifactId**>     <**version**>6.14.3</**version**>     <**scope**>test</**scope**> </**dependency**> <**dependency**> <**groupId**>com.aventstack</**groupId**> <**artifactId**>extentreports</**artifactId**> <**version**>3.1.5</**version**> </**dependency**>

ExtentManager :

**package com.Reporting.Extent;**

**import java.io.File;**

**import com.aventstack.extentreports.ExtentReports;**

**import com.aventstack.extentreports.reporter.ExtentHtmlReporter;**

**import com.aventstack.extentreports.reporter.configuration.ChartLocation;**

**import com.aventstack.extentreports.reporter.configuration.Theme;**

**public class ExtentManager {**

**private static ExtentReports extent;**

**private static String reportFileName = "Test-Automaton-Report"+".html";**

**private static String fileSeperator = System.getProperty("file.separator");**

**private static String reportFilepath = System.getProperty("user.dir") +fileSeperator+ "TestReport";**

**private static String reportFileLocation = reportFilepath +fileSeperator+ reportFileName;**

**public static ExtentReports getInstance() {**

**if (extent == null)**

**createInstance();**

**return extent;**

**}**

**//Create an extent report instance**

**public static ExtentReports createInstance() {**

**String fileName = getReportPath(reportFilepath);**

**ExtentHtmlReporter htmlReporter = new ExtentHtmlReporter(fileName);**

**htmlReporter.config().setTestViewChartLocation(ChartLocation.BOTTOM);**

**htmlReporter.config().setChartVisibilityOnOpen(true);**

**htmlReporter.config().setTheme(Theme.STANDARD);**

**htmlReporter.config().setDocumentTitle(reportFileName);**

**htmlReporter.config().setEncoding("utf-8");**

**htmlReporter.config().setReportName(reportFileName);**

**//htmlReporter.config().setTimeStampFormat("EEEE, MMMM dd, yyyy, hh:mm a '('zzz')'");**

**extent = new ExtentReports();**

**extent.attachReporter(htmlReporter);**

**//Set environment details**

**extent.setSystemInfo("OS", "Windows");**

**extent.setSystemInfo("AUT", "QA");**

**return extent;**

**}**

**//Create the report path**

**private static String getReportPath (String path) {**

**File testDirectory = new File(path);**

**if (!testDirectory.exists()) {**

**if (testDirectory.mkdir()) {**

**System.out.println("Directory: " + path + " is created!" );**

**return reportFileLocation;**

**} else {**

**System.out.println("Failed to create directory: " + path);**

**return System.getProperty("user.dir");**

**}**

**} else {**

**System.out.println("Directory already exists: " + path);**

**}**

**return reportFileLocation;**

**}**

**}**

ExtentTestManager

**package com.Reporting.Extent;**

**import java.util.HashMap;**

**import java.util.Map;**

**import com.aventstack.extentreports.ExtentReports;**

**import com.aventstack.extentreports.ExtentTest;**

**public class ExtentTestManager {**

**static Map<Integer, ExtentTest> extentTestMap = new HashMap<Integer, ExtentTest>();**

**static ExtentReports extent = ExtentManager.getInstance();**

**public static synchronized ExtentTest getTest() {**

**return (ExtentTest) extentTestMap.get((int) (long) (Thread.currentThread().getId()));**

**}**

**public static synchronized void endTest() {**

**extent.flush();**

**}**

**public static synchronized ExtentTest startTest(String testName) {**

**ExtentTest test = extent.createTest(testName);**

**extentTestMap.put((int) (long) (Thread.currentThread().getId()), test);**

**return test;**

**}**

**}**

TestListener :

**package com.Reporting.Extent;**

**import java.util.Arrays;**

**import org.testng.ITestContext;**

**import org.testng.ITestListener;**

**import org.testng.ITestResult;**

**import com.Reporting.Extent.ExtentTestManager;**

**import com.aventstack.extentreports.Status;**

**public class TestListener implements ITestListener {**

**public void onStart(ITestContext context) {**

**System.out.println("\*\*\* Test Suite " + context.getName() + " started \*\*\*");**

**}**

**public void onFinish(ITestContext context) {**

**System.out.println(("\*\*\* Test Suite " + context.getName() + " ending \*\*\*"));**

**ExtentTestManager.endTest();**

**ExtentManager.getInstance().flush();**

**}**

**public void onTestStart(ITestResult result) {**

**System.out.println(("\*\*\* Running test method " + result.getMethod().getMethodName() + "..."));**

**ExtentTestManager.startTest(result.getMethod().getMethodName());**

**}**

**public void onTestSuccess(ITestResult result) {**

**System.out.println("\*\*\* Executed " + result.getMethod().getMethodName() + " test successfully...");**

**ExtentTestManager.getTest().log(Status.PASS, "Test passed");**

**}**

**public void onTestFailure(ITestResult result) {**

**System.out.println("\*\*\* Test execution " + result.getMethod().getMethodName() + " failed...");**

**String excepionMessage = Arrays.toString(result.getThrowable().getStackTrace());**

**String errMessage="<b>" + "<font color=" + "red>" + "Exception Occured:Click to see"**

**+ "</font>" + "</b >" + excepionMessage.replaceAll(",", "<br>")**

**+ " \n";**

**ExtentTestManager.getTest().log(Status.FAIL, "Test Failed"+"<br>"+errMessage);**

**}**

**public void onTestSkipped(ITestResult result) {**

**System.out.println("\*\*\* Test " + result.getMethod().getMethodName() + " skipped...");**

**ExtentTestManager.getTest().log(Status.SKIP, "Test Skipped");**

**}**

**public void onTestFailedButWithinSuccessPercentage(ITestResult result) {**

**System.out.println("\*\*\* Test failed but within percentage % " + result.getMethod().getMethodName());**

**}**

**}**

TestCase2 :

**package** com.Reporting.Extent;

**import** org.testng.Assert;

**import** org.testng.SkipException;

**import** org.testng.annotations.Test;

**import** com.aventstack.extentreports.Status;

**public** **class** TestCase2 {

@Test

**public** **void** doLogin() {

ExtentTestManager.*getTest*().log(Status.***INFO***, "doLogin test started");

System.***out***.println("Executing Login Test");

ExtentTestManager.*getTest*().log(Status.***INFO***, "doLogin test completed");

}

@Test

**public** **void** doUserReg() {

ExtentTestManager.*getTest*().log(Status.***INFO***, "doUserReg test started");

Assert.*fail*("User Reg Test Failed");

ExtentTestManager.*getTest*().log(Status.***INFO***, "doUserReg test completed");

}

@Test

**public** **void** isSkip() {

ExtentTestManager.*getTest*().log(Status.***INFO***, "Skipping the test case");

**throw** **new** SkipException("Skipping the test case");

}

}

### **How to capture screenshots in Extent Report**

By capturing screenshots, testers can better identify what went wrong when the software acted erroneously during a test. Capture screenshots only when a test fails, since they consume a lot of memory.

Try capturing screenshots with the code below

test.log(LogStatus.FAIL,test.addScreenCapture(capture(driver))+ "Test Failed");

public static String capture(WebDriver driver) throws IOException {

File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

File Dest = new File("src/../BStackImages/" + System.currentTimeMillis()

+ ".png");

String errflpath = Dest.getAbsolutePath();

FileUtils.copyFile(scrFile, Dest);

return errflpath;

}

**getScreenShotAs():** Captures screenshot of the current WebDriver instance and stores it in different output forms.

File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

This method returns a file object to be stored onto a file variable. Casting the web driver instance to Take Screenshot is necessary to use the method.

File Dest = new File("src/../BStackImages/" + System.currentTimeMillis()+ ".png");

This statement creates a folder named ‘**BStackImages**’ within the ‘**src**’ folder and stores the file name as the current system time.

String errflpath = Dest.getAbsolutePath();

FileUtils.copyFile(scrFile, Dest);

returnerrflpath;

These statements copy all error images to the destination folder.

Use the log method because it uses the **addScreenCapture** method of **Extent Test** class to get a screenshot and add it to the **Extent Report**.

test.log(LogStatus.FAIL,test.addScreenCapture(capture(driver))+ "Test Failed");

### **Benefits of using Extent Reports**

* They can be integrated with TestNG and JUnit
* If required, screenshots can be captured and displayed for each step in a test
* They allow testers to track multiple test case runs in a single test suite
* They show the time needed for test execution
* They can be customized to graphically represent each step in a test.