4. Demonstrate how extent reports are generated

Step 1: Generating Extent Reports

- Open Eclipse.

- Create a TestNG project in Eclipse.
- Extent Reports jar file is already present in your practice lab in /home/ubuntu/libs directory. Refer QA to QE lab guide for Phase 2 for more information.
- Add the Extent Reports jar file to your project.
- Create a java class, say **ExtentReportsClass** and add the following code to it.
- Sample code:

```
import java.io.File;
 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.Test;

import com.relevantcodes.extentreports.ExtentReports;
import com.relevantcodes.extentreports.ExtentTest;
import com.relevantcodes.extentreports.LogStatus;

public class ExtentReportsClass{
 ExtentReports extent;
 ExtentTest logger;
 @BeforeTest
 public void startReport(){
 //ExtentReports(String filePath,Boolean replaceExisting)
 //filepath - path of the file, in .htm or .html format - path where your report needs to generate.
//replaceExisting - Setting to overwrite (TRUE) the existing file or append to it
 //True (default): the file will be replaced with brand new markup, and all existing data will be lost. Use this option to create a brand new report
 //False: existing data will remain, new tests will be appended to the existing report. If the supplied path does not exist, a new file will be created.
 extent = new ExtentReports (System.getProperty("user.dir") +"/test-output/STMExtentReport.html", true);
 //extent.addSystemInfo("Environment","Environment Name")
 extent
            .addSystemInfo("Host Name", "SoftwareTesting")
```

```
            .addSystemInfo("Environment", "Automation Testing")
            .addSystemInfo("User Name", "TestEngineer");
            //loading the external xml file (i.e., extent-config.xml) that was placed under the base directory
            //You could find the xml file below. Create xml file in your project and copy paste the code
mentioned below
            extent.loadConfig(new File(System.getProperty("user.dir")+"\\extent-config.xml"));
 }
 @Test
 public void passTest(){
 //extent.startTest("TestCaseName", "Description")
 //TestCaseName – Name of the test
 //Description – Description of the test
 //Starting test
 logger = extent.startTest("passTest");
 Assert.assertTrue(true);
 //To generate the log when the test case is passed
 logger.log(LogStatus.PASS, "Test Case Passed is passTest"); }
@Test
 public void failTest(){
 logger = extent.startTest("failTest");
 Assert.assertTrue(false);
 logger.log(LogStatus.PASS, "Test Case (failTest) Status is passed");
 }

 @Test
 public void skipTest(){
 logger = extent.startTest("skipTest");
 throw new SkipException("Skipping - This is not ready for testing ");
 }
 @AfterMethod
 public void getResult(ITestResult result){
 if(result.getStatus() == ITestResult.FAILURE){
 logger.log(LogStatus.FAIL, "Test Case Failed is "+result.getName());
 logger.log(LogStatus.FAIL, "Test Case Failed is "+result.getThrowable());
 }else if(result.getStatus() == ITestResult.SKIP){
 logger.log(LogStatus.SKIP, "Test Case Skipped is "+result.getName());
 }
 // ending test
 //endTest(logger) : It ends the current test and prepares to create a HTML report
 extent.endTest(logger);
 }
 @AfterTest
 public void endReport(){
 //writing everything to document
 //flush() - to write or update test information to your report.
            extent.flush();
            //Call close() at the very end of your session to clear all resources.
```

```
        //If any of your tests ended abruptly causing any side-affects (not all logs sent to
ExtentReports, information missing), this method will ensure that the test is still appended to the report
with a warning message.
        //You should call close() only once, at the very end (in @AfterSuite for example) as it closes the
underlying stream.
        //Once this method is called, calling any Extent method will throw an error.
        //close() - To close all the operations
        extent.close();
    }
}
```

- Code explanation:

i. Imported two classes *ExtentReports* and *ExtentTest*.

ExtentReports: By using this class, we set the path where our reports need to be generated.

ExtentTest: By using this class, we could generate the logs in the report.

ii. Took three methods with @Test annotation such as *passTest*, *failTest* and *skipTest* and a method *startTest* with @BeforeTest annotation and another method *endReport* with @AfterMethod annotation

iii. The used object of *ExtentReports* class (i.e., *extent*) in the *startReport* method which was assigned to @BeforeTest annotation to generate the HTML report in the required path.

iv. The used object of *ExtentTest* class (i.e., *logger*) in the remaining methods to write logs in the report.

v. Used *ITestResult* class in the @AfterMethod to describes the result of a test

Step 2: Describing Extent-config.xml

By using this external XML file (extent-config.xml), we could change the details, such as Report Theme, Report Title, and Document Title.

We use the extent object and use loadConfig() method to load this XML file.

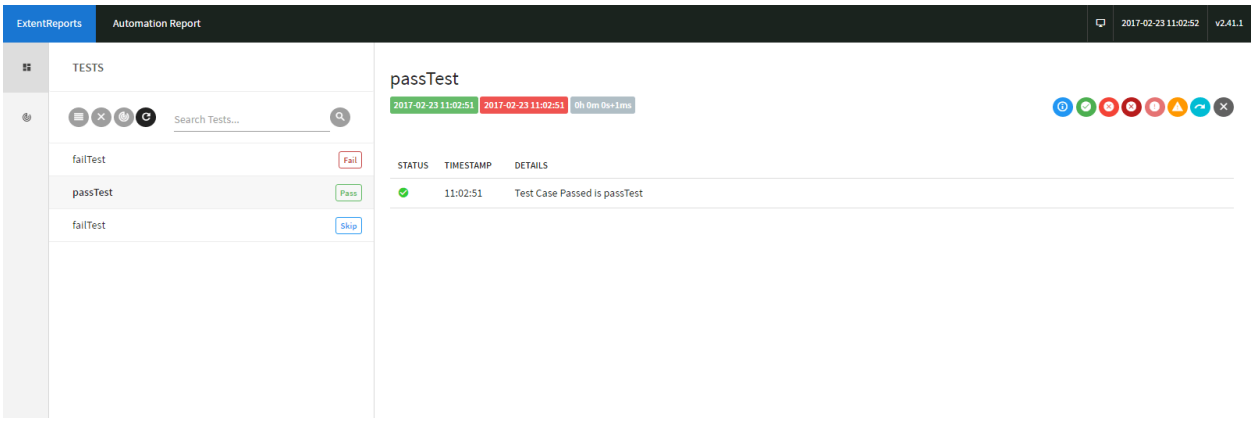
```
<?xml version="1.0" encoding="UTF-8"?>
<extentreports>
  <configuration>
    <!-- report theme -->
    <!-- standard, dark -->
    <theme>standard</theme>
    <!-- document encoding -->
    <!-- defaults to UTF-8 -->
    <encoding>UTF-8</encoding>
    <!-- protocol for script and stylesheets -->
    <!-- defaults to https -->
    <protocol>https</protocol>
     <!-- title of the document -->
    <documentTitle>ExtentReports 2.0</documentTitle>
     <!-- report name - displayed at top-nav -->
    <reportName></reportName>
    <!-- report headline - displayed at top-nav, after reportHeadline -->
    <reportHeadline>Automation Report</reportHeadline>
    <!-- global date format override -->
    <!-- defaults to yyyy-MM-dd -->
    <dateFormat>yyyy-MM-dd</dateFormat>
    <!-- global time format override -->
    <!-- defaults to HH:mm:ss -->
    <timeFormat>HH:mm:ss</timeFormat>
    <!-- custom javascript -->
    <scripts>
     <![CDATA[
       $(document).ready(function() {
     });
     ]]>
    </scripts>
    <!-- custom styles -->
    <styles>
     <![CDATA[
        ]]>
    </styles>
  </configuration>
</extentreports>
```

- Console Output:

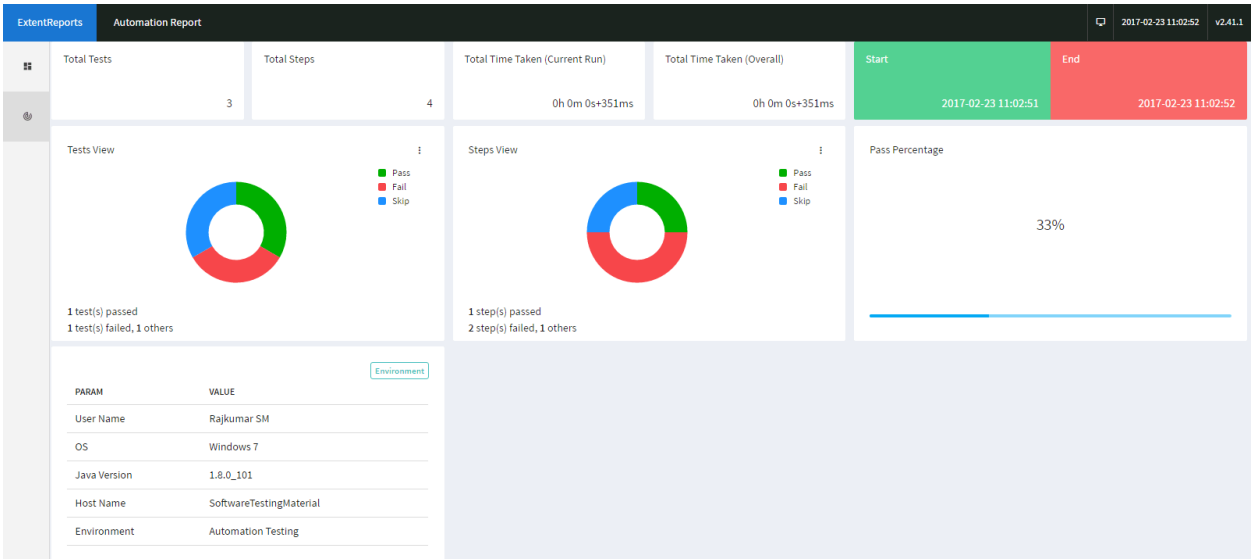
```
1 ===============================================
2 Default suite
3 Total tests run: 3, Failures: 1, Skips: 1
4 ===============================================
```

Refresh the project after execution of the above ExtentReportsClass.java file. You can find an HTML file named **STMExtentReport.html** in your test-output folder. Copy the location of the STMExtentReport.html file and open it by using any browser. Once you open the report, you will see rich HTML test results as shown below.

Step 3: Fetching Test Summary Report



Graphical Report with PIE Charts:



Step 4: Pushing the code to your GitHub repositories
Open your command prompt and navigate to the folder where you have created your files.
cd <folder path>
Initialize your repository using the following command:
git init
Add all the files to your git repository using the following command:
git add .
Commit the changes using the following command:
git commit . -m “Changes have been committed.”
Push the files to the folder you initially created using the following command:
git push -u origin master