For Uniathena Short course Automation

Framework Manual



## Contents

[1. Document Version 3](#_Toc610213433)

[2. About Framework 4](#_Toc1622754006)

[3. Folder Structure 5](#_Toc584261942)

[4. Framework Components 6](#_Toc1395225110)

[5. Handling Database 7](#_Toc1174482177)

[6. Setting up and updating test data 8](#_Toc1999749262)

[7. How to start scripting? 9](#_Toc1666770884)

[8. How to triger the test? 11](#_Toc1641555920)

[9. How to verify the result? 12](#_Toc907934431)

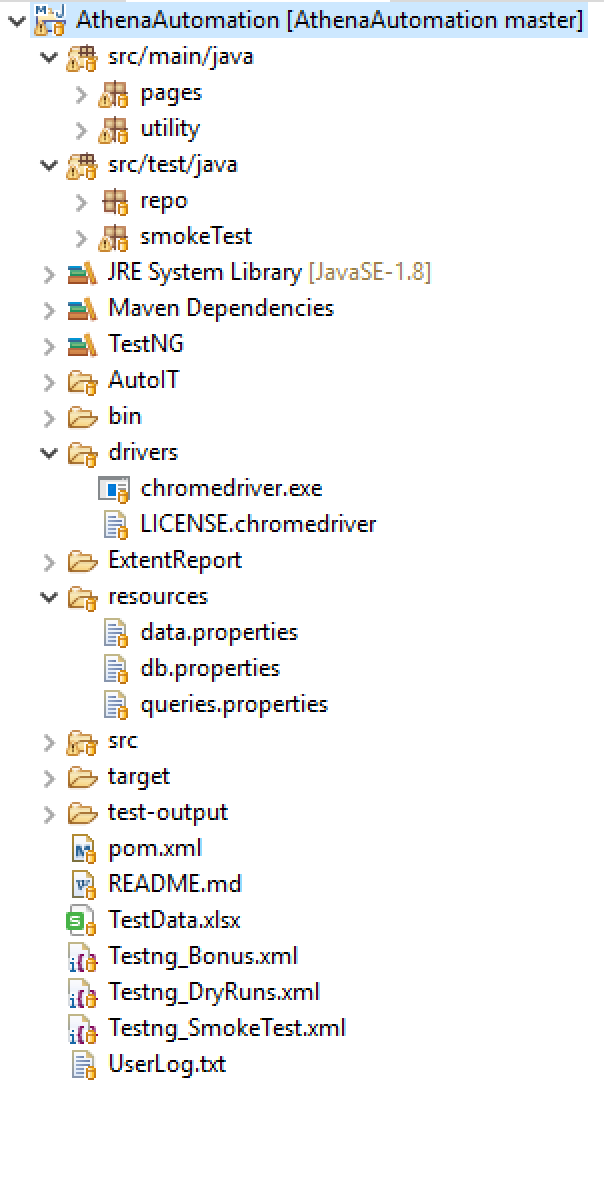
# Document Version

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Updated by** | **Date** | **Comments** |
| 1.0 | Parvathi P | 15 March 2023 | First version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# **About** Framework

Our automation framework is constructed on top of TestNG, and the scripts are generated in Java language and is automated using Selenium. The dependency management is handled using Maven. Also, we have a rich HTML report formed with extent reporting and troubleshooting can be achieved using the auto generated TestNG. The page object modelling is implement in order to ensure the re-usability and easy maintenance. The desktop components are handled using AutoIT.

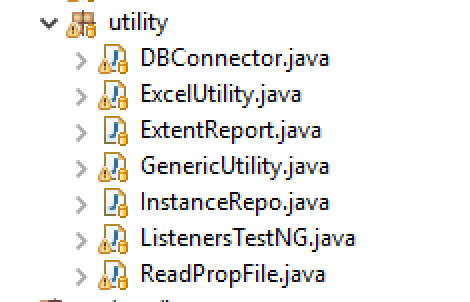
# **Folder** Structure



The main folders containing the script components are as below:

* **src/main/java** - contains framework functions and POM classes
* **pages -** follows the page object model and for each screen in the application is mapped with a class in the ‘pages’ folder
* **utility** is the folder that has all the framework related classes and components
* **src/test/java** - This folder contains the test scripts
* **drivers** - This folder contains the executables of the browsers/drivers used for execution
* **resources -** It contains the property files that has values like urls, credentials, queries etc.
* **pom.xml** - This has all the dependencies that are used in the framework
* **TestData.xlsx** - All the test data used for automation are maintained in this spreadsheet file. multiple sheets can be added to organize the data based on the test case
* **UserLog.txt** - A text file that collects all the user details created through automation.
* **Testng\_\*\*\*\*\*.xml** - These are the xml files linked to TestNG. The tests are triggered from here. Required class named to be executed needs to be updated in this file

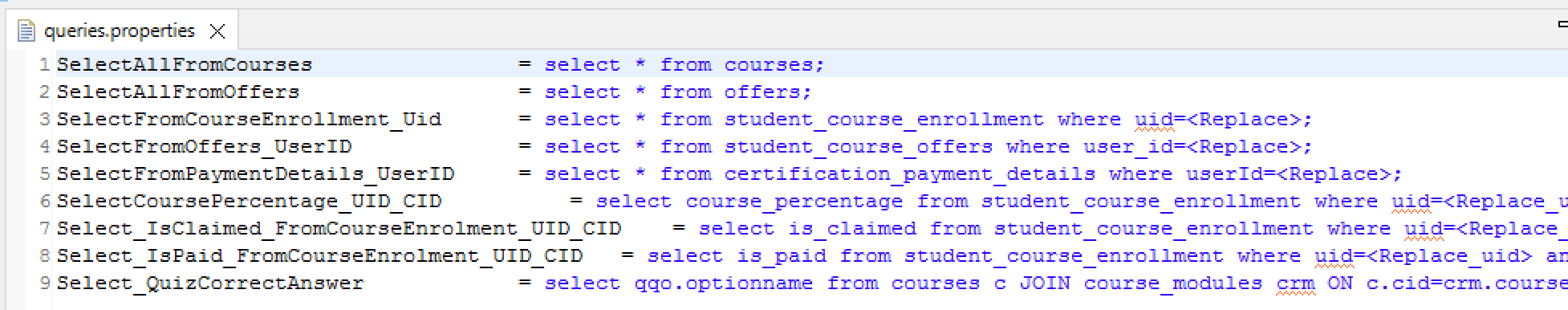
# **Framework** Components

A number of components are available in the framework built. Below are the classes created specifically for implementing some requirements.

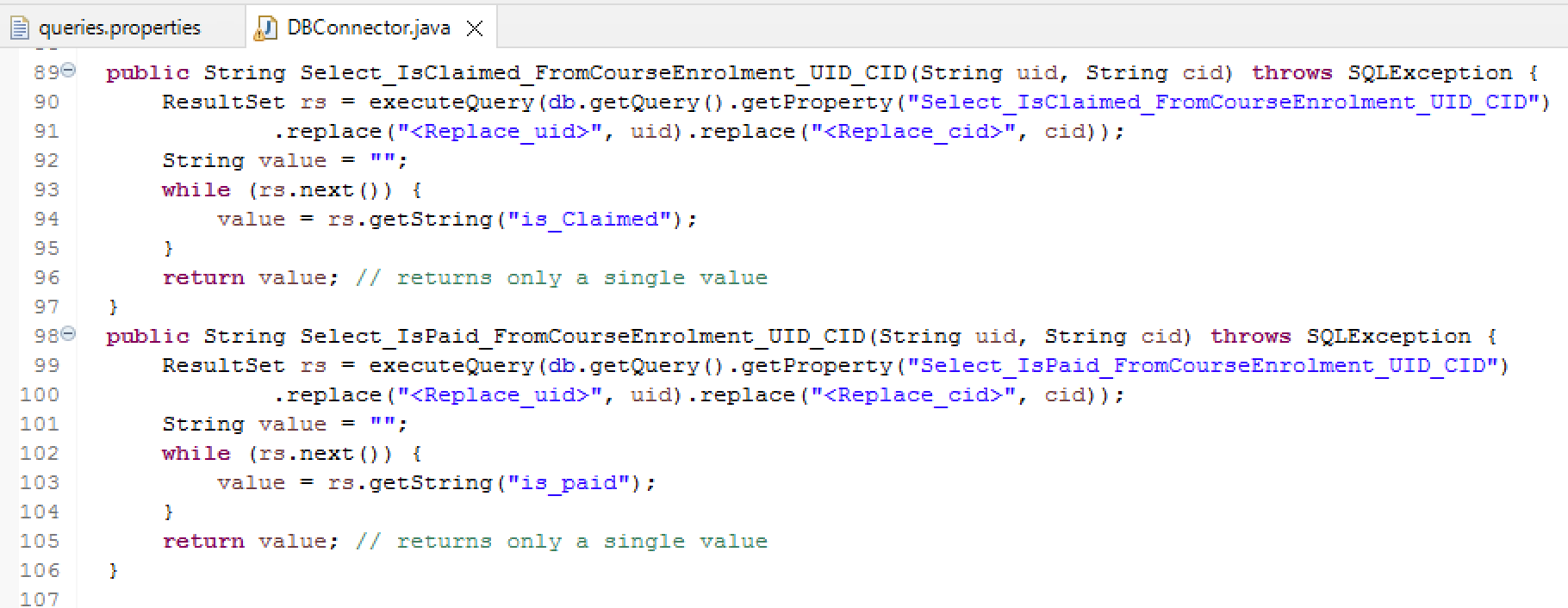
* **DBConnector -** This is used to initiate and establish connection. Multiple methods are implemented here to execute the query by reading the data from the queries.property file
* **ExcelUtility -** Test data related methods are implemented in this class
* **GenericUtility** - All generic and common operations for enabling the automation are implemented here. For eg: click, sendkeys, randomNumberGeneration etc.
* **InstanceRepo -**  This is implemented to retrieve the genericUtility and other class object details on the runtime.
* **ListenersTestNG -** Implementing this in order to capture the failure in the HTML reports.
* **ReadPropFile -** This is used to read the content from the \*.properties file implemented in the project

# **Handling Database**

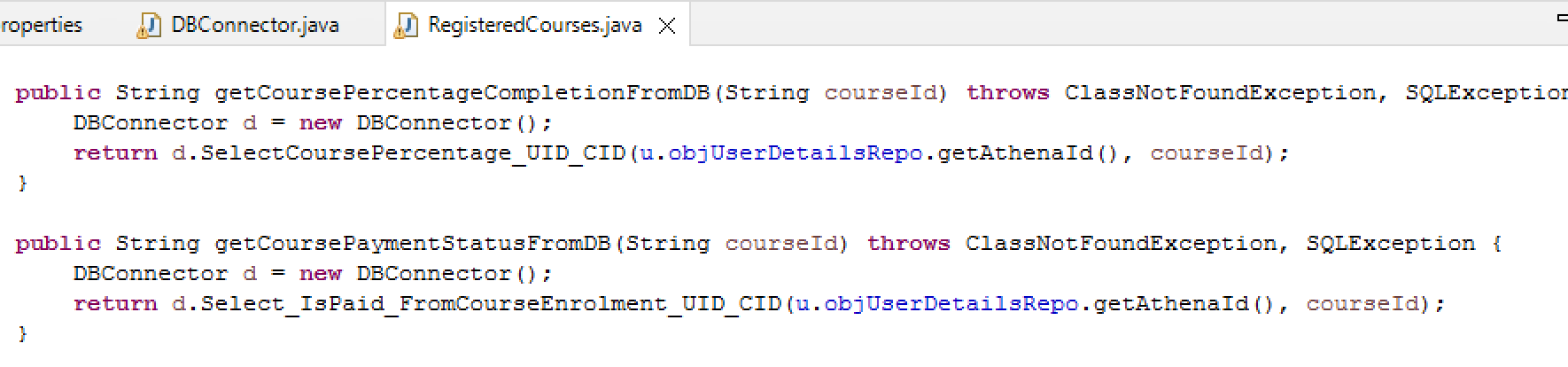
**Step1:** In queries.properties file, copy the query with proper naming of the variable. Replace the dynamic values in the form of ‘*<Replace\_\*\*\*\*\*\*>*’ where \* indicates the value name.



**Step 2:** In *DBConnector.java* file, add methods to call the queries mentioned in *queries.properties* file and replace the dynamic values with the parameters.

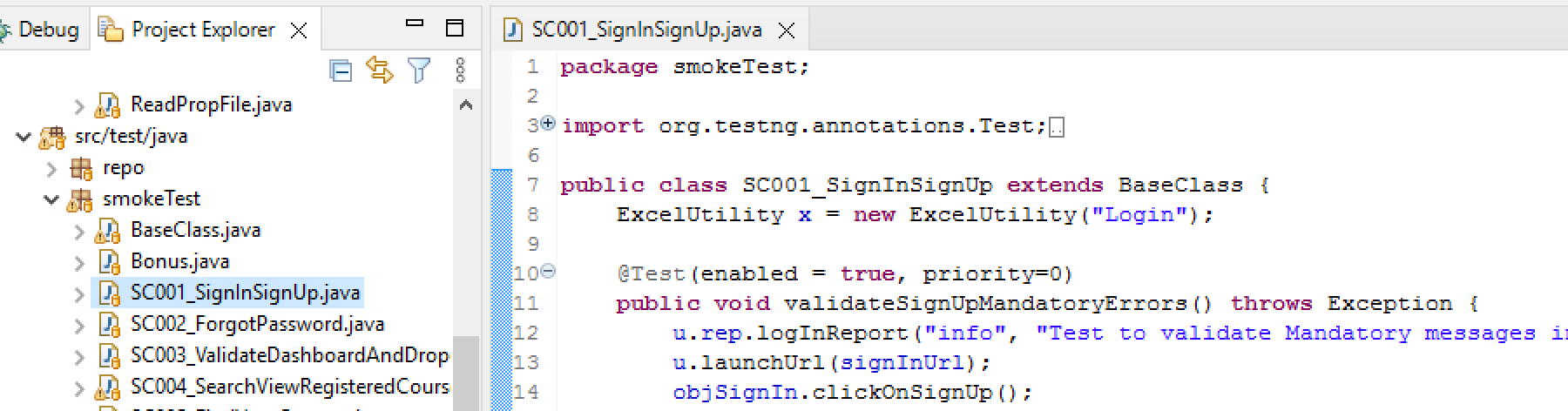


**Step 3:** Call the methods derived in step 2 at the required locations

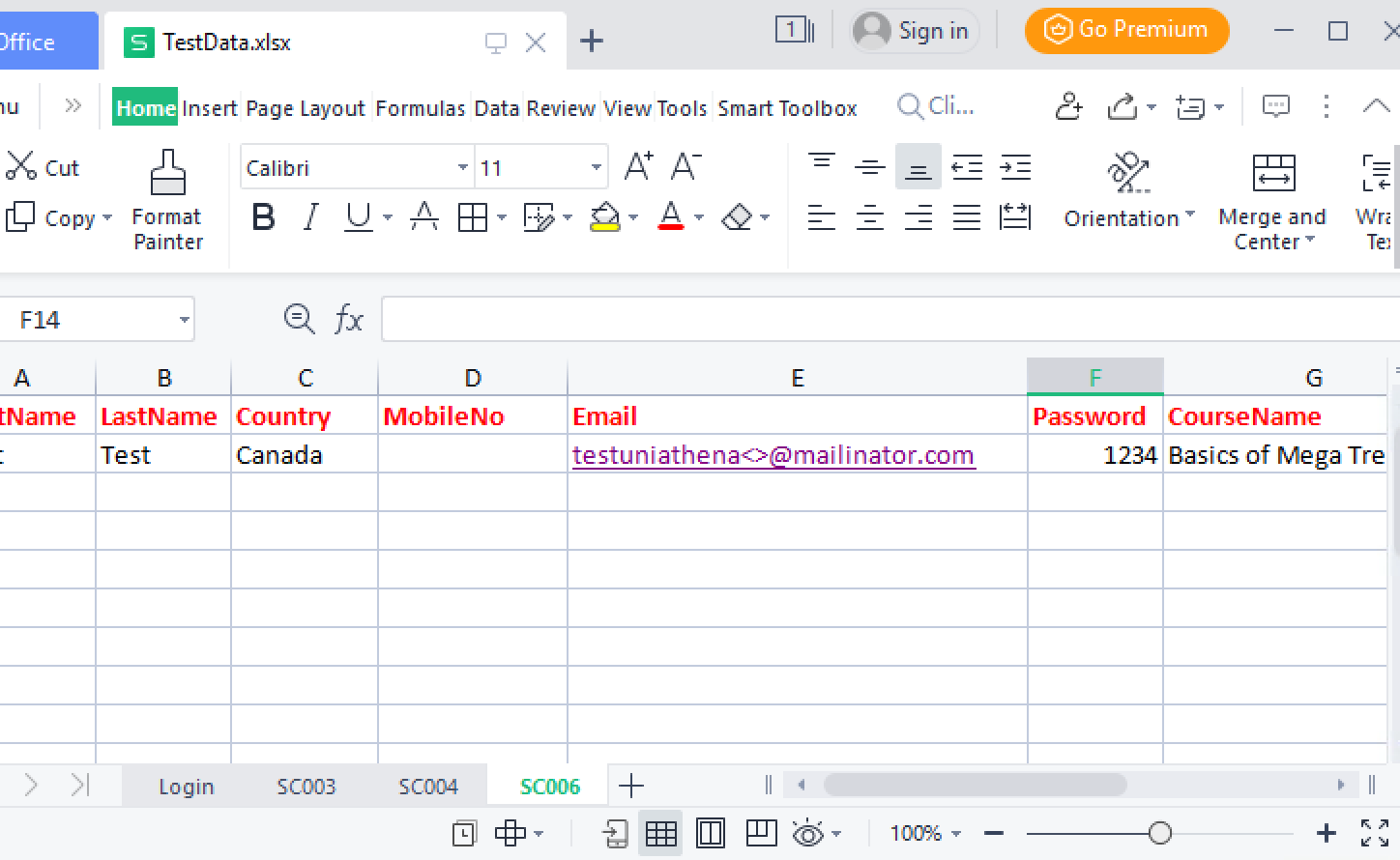


# **Setting up and updating test data**

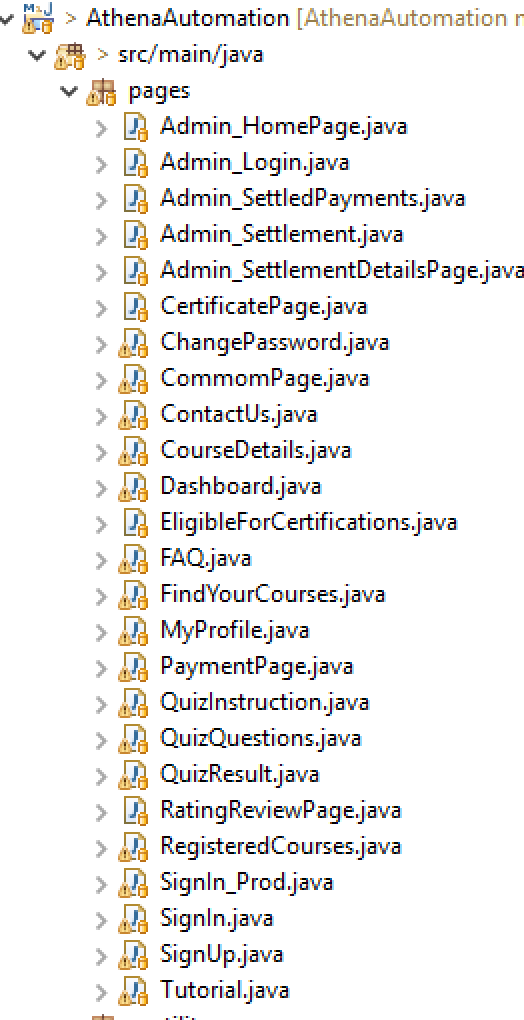
Test data can be maintained in the spreadsheet file available in the project folder named as *Testdata.xlsx*. In order to use the test data set up in the spreadsheet, we need to create an object for ExcelUtility by passing the sheet name as the parameter.



As mentioned in the section ‘*3. Folder Structure*’ test data of different test scripts or test cases can be kept on different sheets. This can be altered logically by linking multiple test scripts to a single sheet in case of avoiding the duplicates.



# **How to start scripting?**

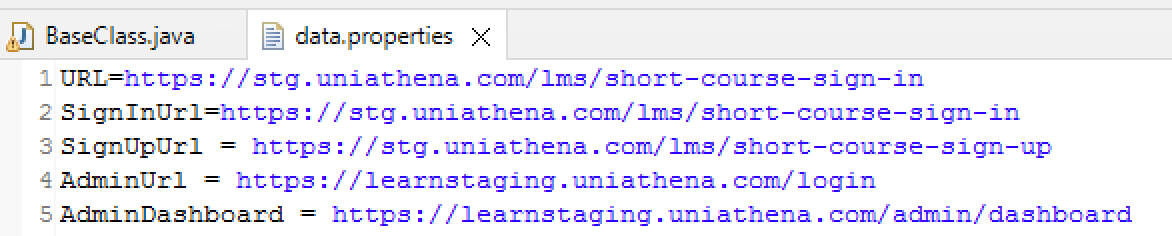
**Step 1:** Check for the available classes in ‘*src/main/java/pages/*‘. If the required page is not implemented, create a class for the page you are implementing the automated operations under ‘*src/main/java/pages/*‘

**Step 2:** If you are creating a new page class, make sure to create an

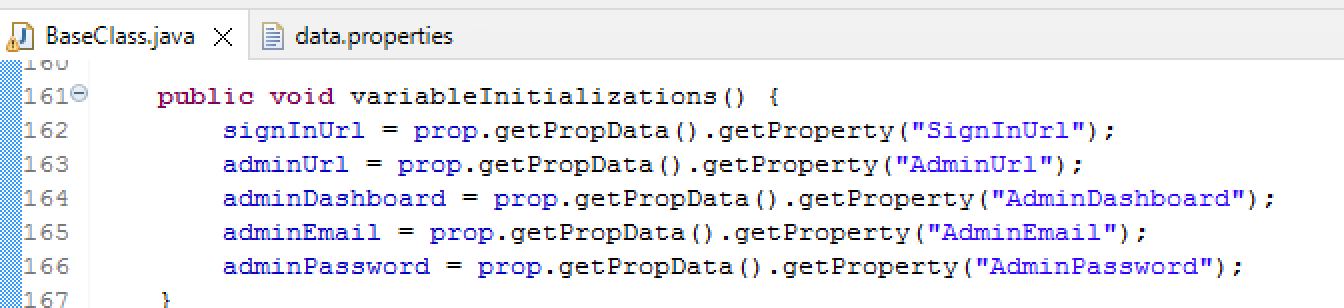
object in *BaseClass.java* available under ‘*src/test/java/*’



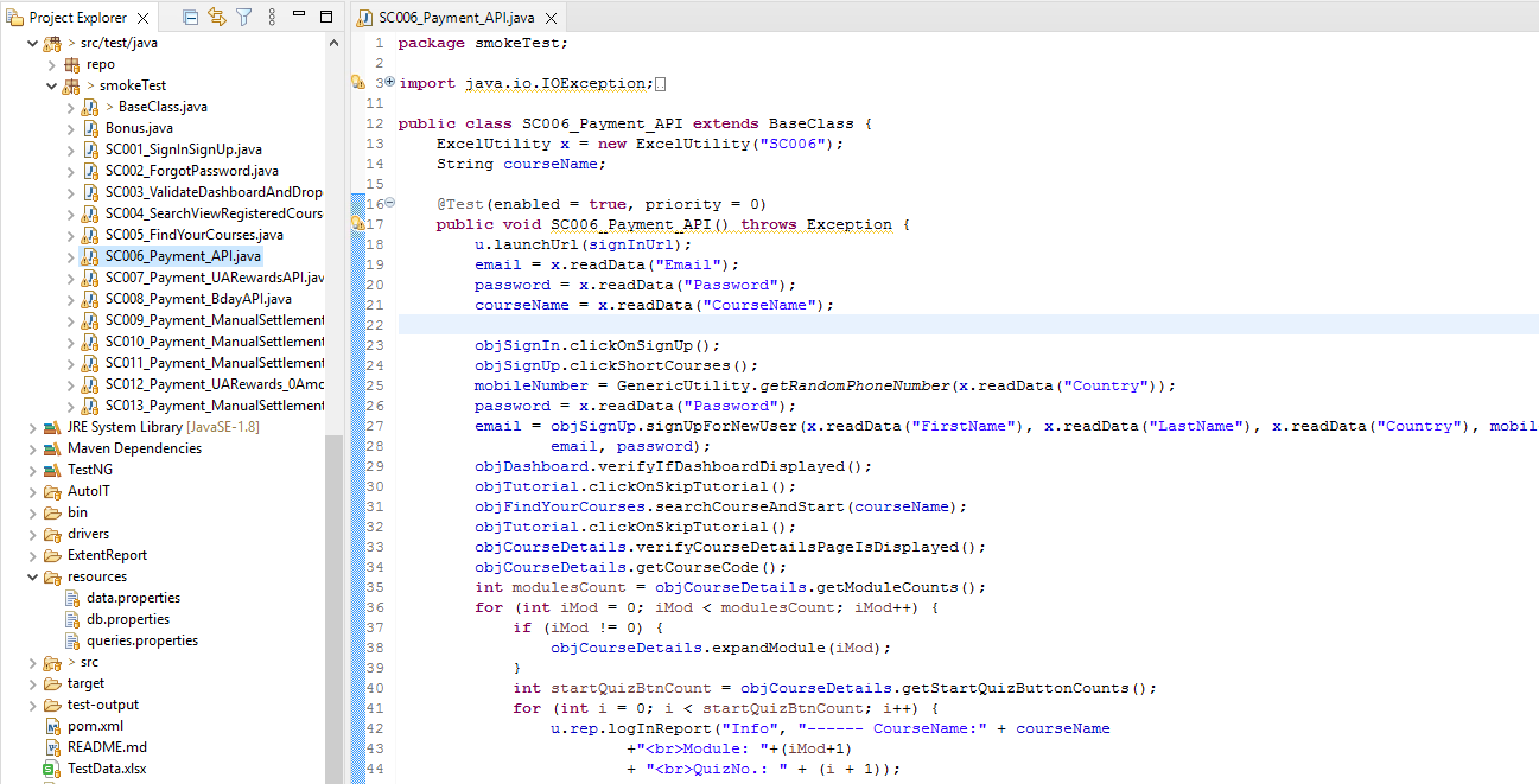
**Step 3:** In case of any new URL addition, add the values in ‘*data.properties*’ file



And make sure to create a variable for the URL in ‘*BaseClass.java*’



**Step 4:** Later make use of the methods implemented in classes of pages folder to complete the test scripting in *src/test/java*



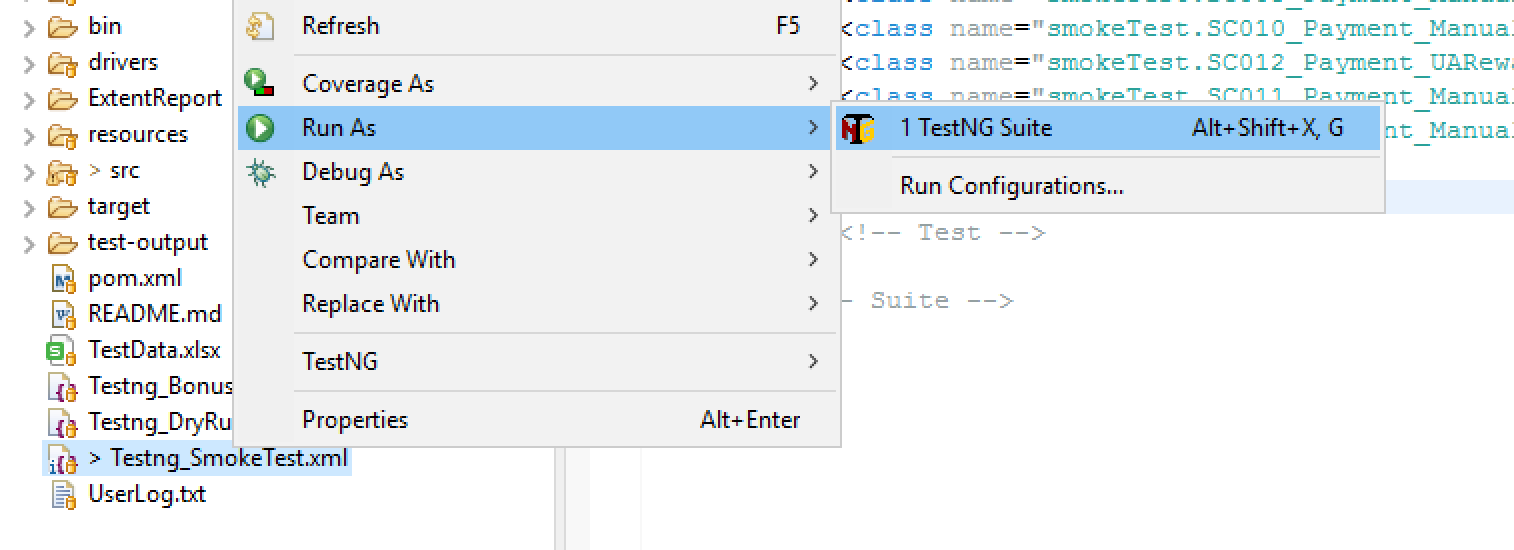
# **How to trigger the test?**

**Step 1:** Pick the required *Testng\_\*\*\*\*.xml* and update the class names required to be executed under the tag <class name =”---------”>

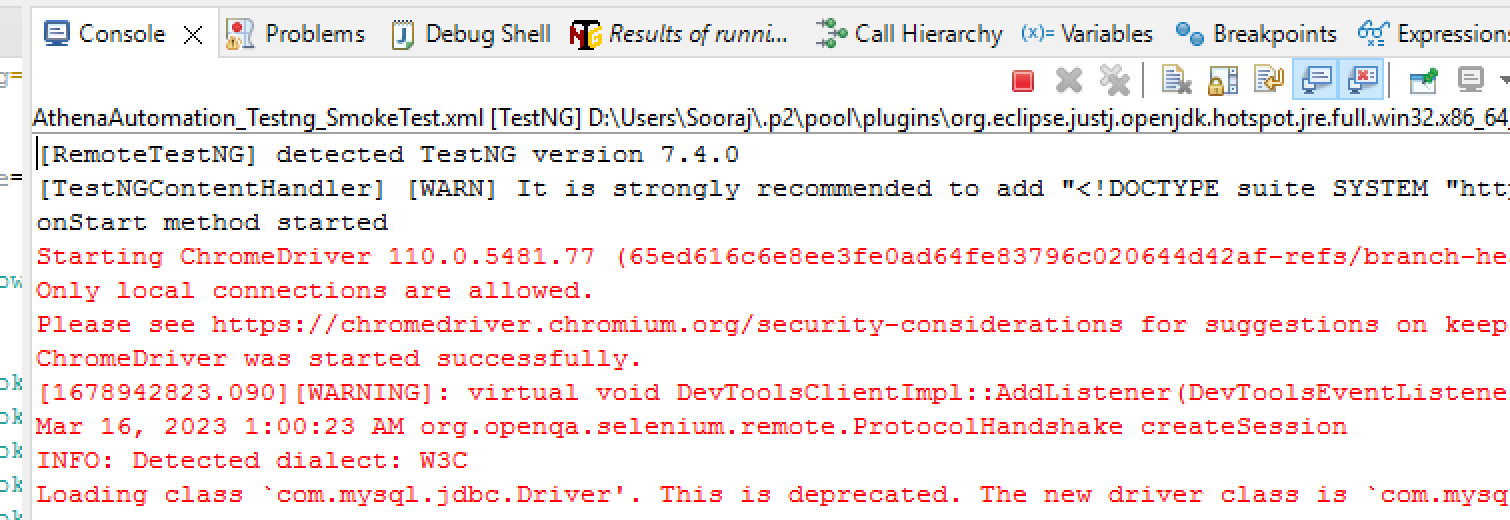


**Step 2:** Update the browser to be launched for execution under the tag <parameter>. Values can be “Firefox”, “Chrome”, “Edge”; which is not case sensitive. Make sure the executable of the driver selected is available under the folder *driver.* If not download the .exe from the selenium official page.

**Step 3:** Right click on the xml file and select *Run As > TestNG Suite*



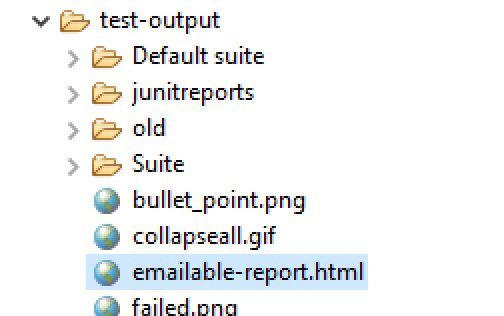
**Step 4:** Confirm you are getting a successful launch of the driver by verifying the console



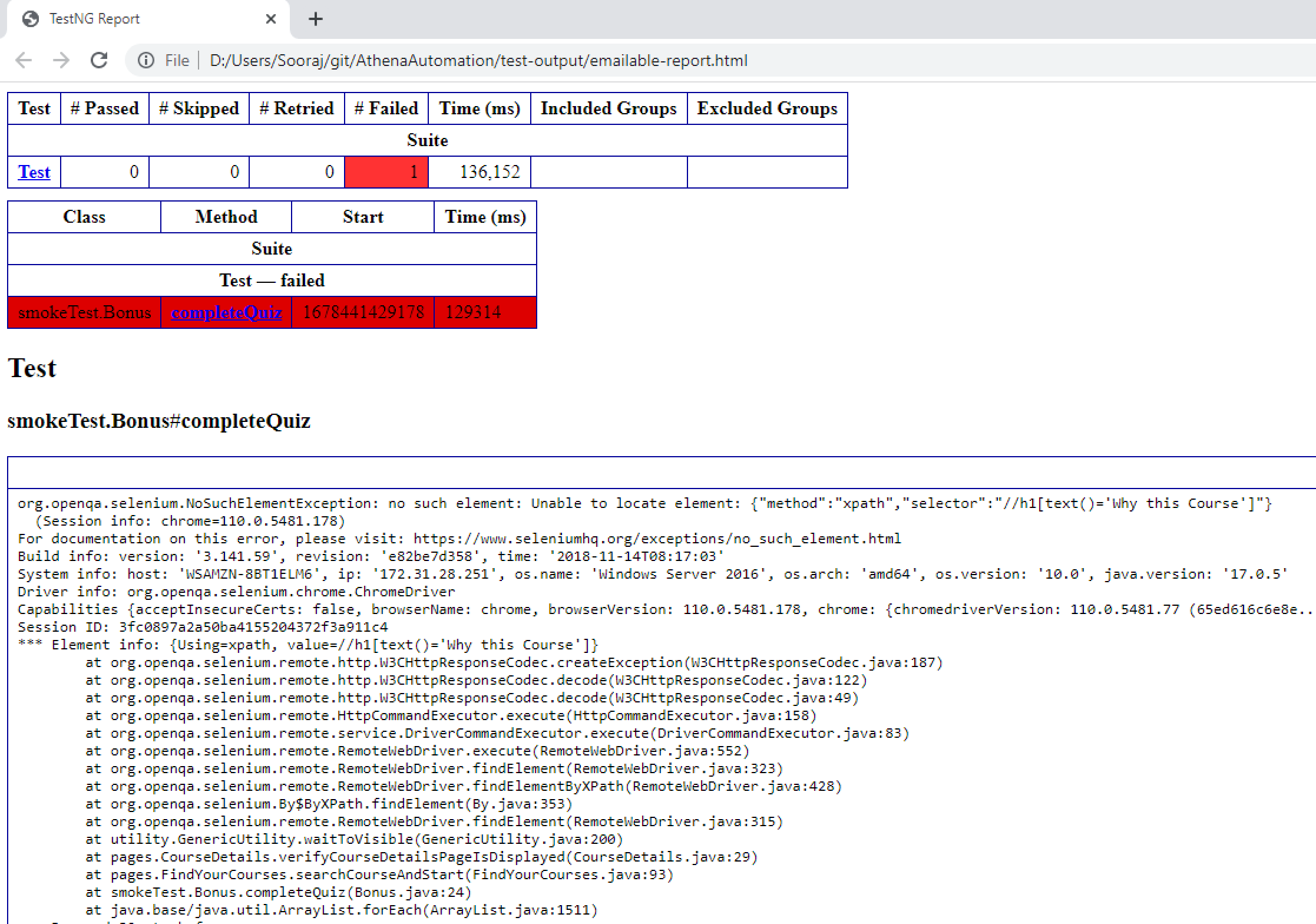
# **How to verify the result?**

We can make use of 2 different report types available with the framework:

1. TestNG Report
2. Extent Report



TestNG report can be viewed from the test-output folder that gets generated/updated after every run



While the Extent Report is a rich HTML report that includes screenshots and step details. New folders with random numbers are generated for each run and the reports are never over written.

