Day 24- Task  POM and Testng

**Project Report url:** [file:///C:/Users/HP/Eclipse%20workspace%20new/Pom/report/Demoreport.html](C://Users/HP/Eclipse%20workspace%20new/Pom/report/Demoreport.html)

Task Description:

Develop a Page Object Model (POM) framework using Selenium to automate test for given requirements:

1. Identify the web pages on https://www.demoblaze.com/ and create a separate Java class for each page.

2. Write test cases for Signup & Login Page the website.

3. Define web elements using the @FindBy annotation from the Selenium WebDriver library.

4. Create methods to perform actions on each web element, such as clicking a button or entering text into a text field.

5. Use the PageFactory class to initialize the web elements.

6. Use assertions to verify that the expected results are achieved.

7. Configure TestNG with the Java project.

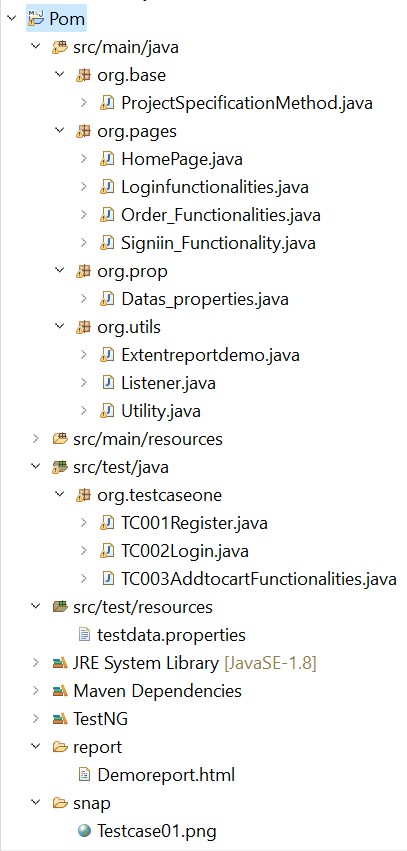
8. Create test suites and add test classes to it.

9. Generate test reports

1. Page Object Model

* The Page Object Model (POM) is a design pattern used in software testing, particularly in the context of automated testing of web applications.it crate repository for strong all the webelement
* In real time work enviranment  every page has one pojoclass (plain old java object class)

In pojo class we stored webelement as  each private object.and also we used getters and setters to get the element and set the values for the private web element

 Org.base:

* base package contains ProjectSpecificationmethod

this class control work flow of the project this is the parent class for all other child classes in project

* but ProjectSpecificationclass is a child class of utils class

Org.pages:

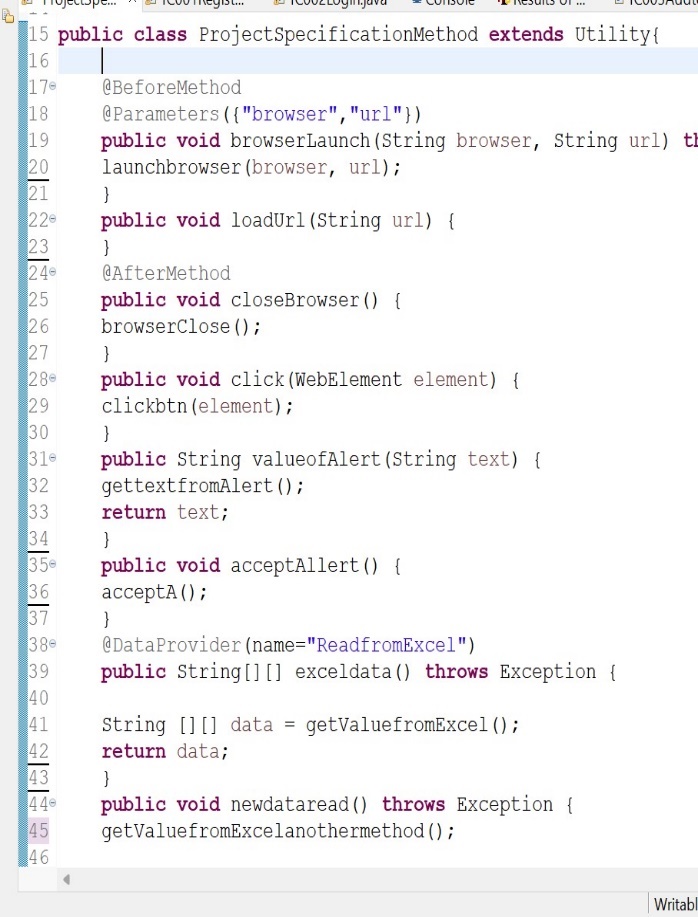
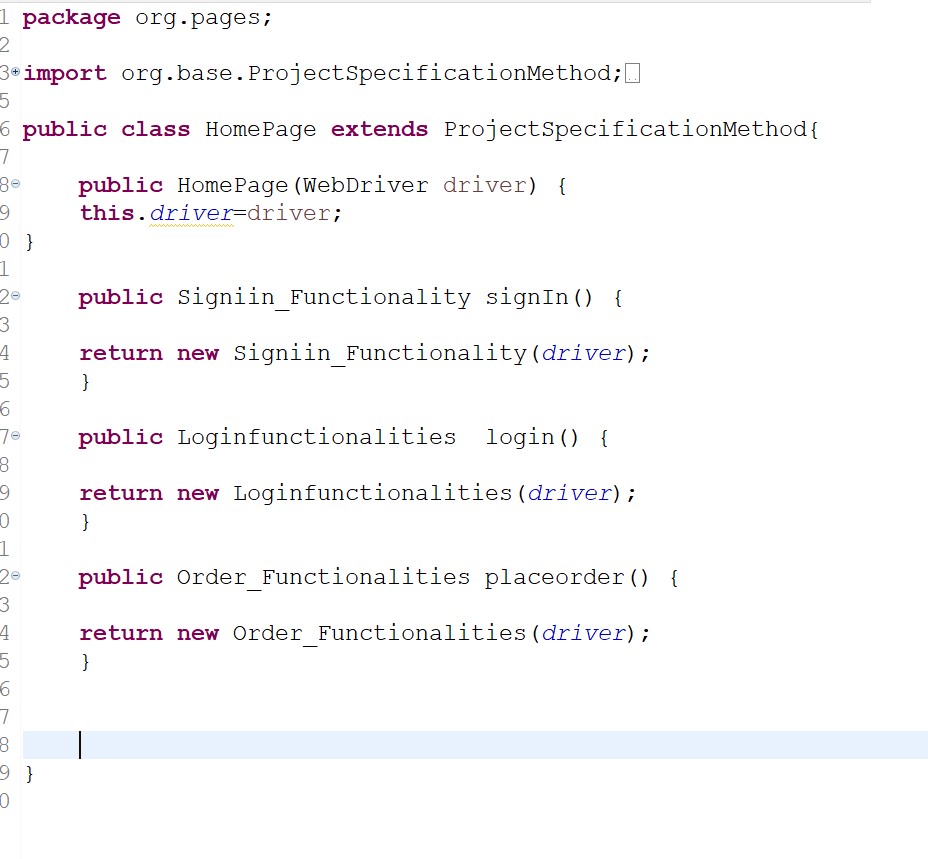
* Pages package contains all the page class in project like if any project has multiple pages login page, signin page, function page, conclude page all pages convert to classes and all that classes will reside pages packages.

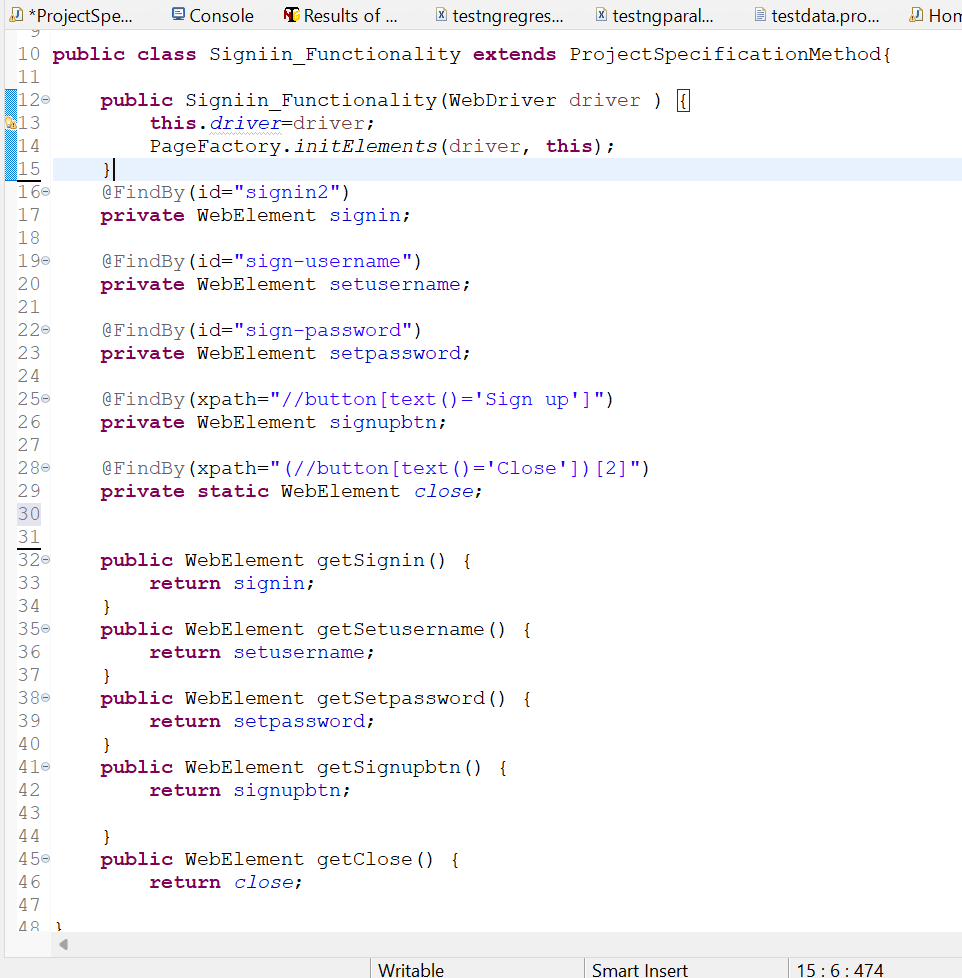
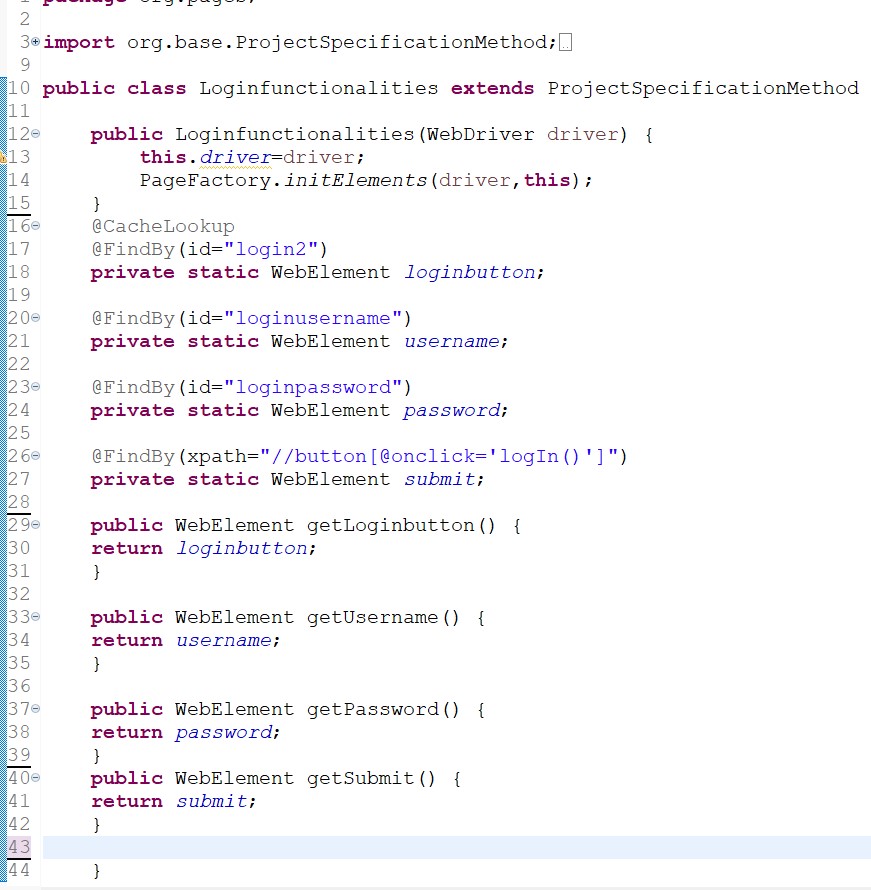
Org.Utils:

* Org.utils package contains utility class, listener class and extent report class.
* utility class is the base class of functions all the functions will reside utility class so when ever we want the particular functions are method we can get it from utils class so our code will appear clean.
* Extent Report Class contains code for generating the extent report of the project.
* Listener class is used to manage the work flow of the testng.xml file

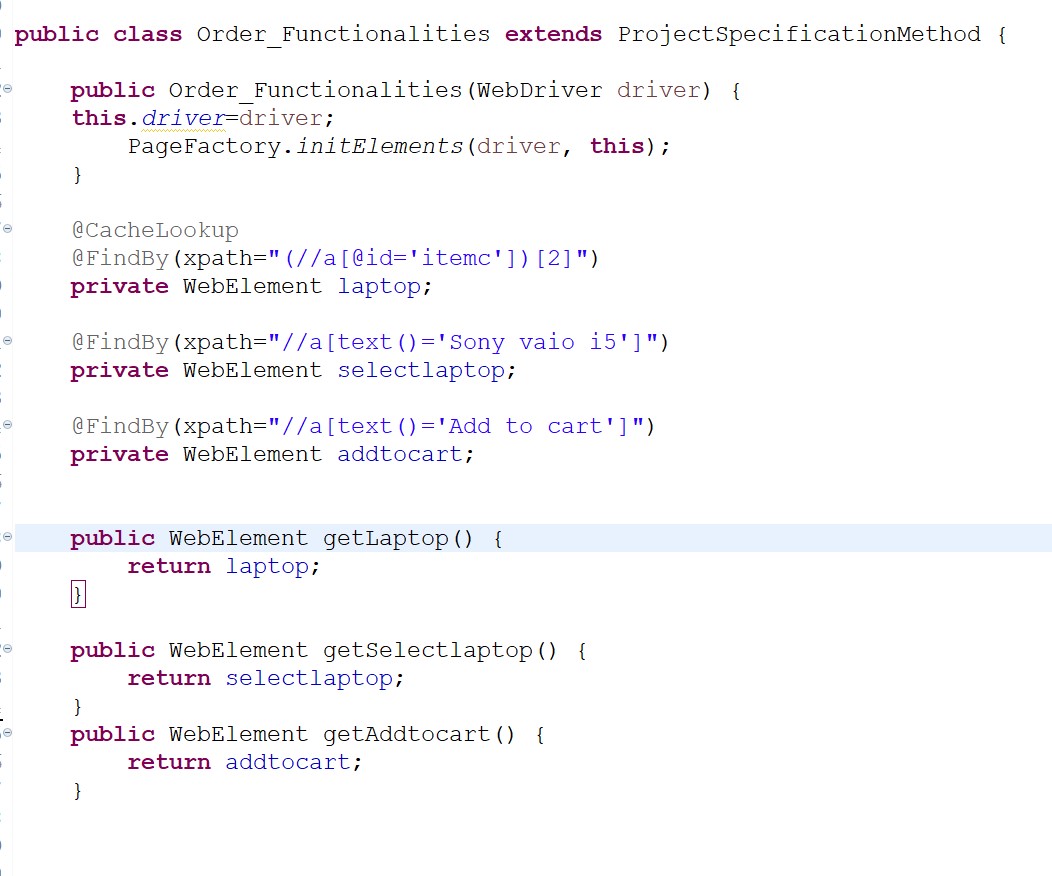
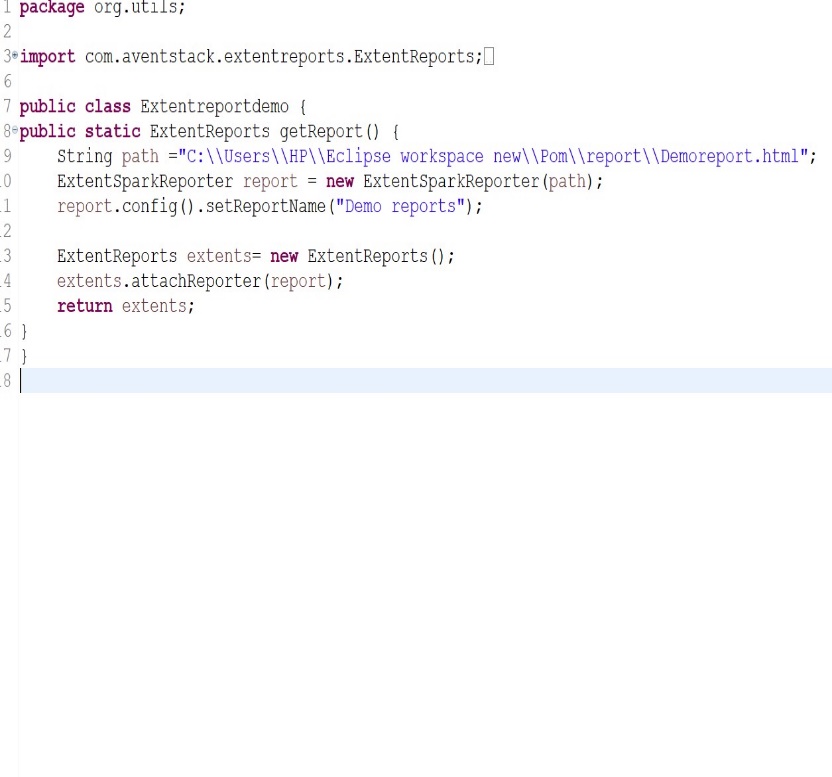
1. Here we will see what classes will actually has with pictures

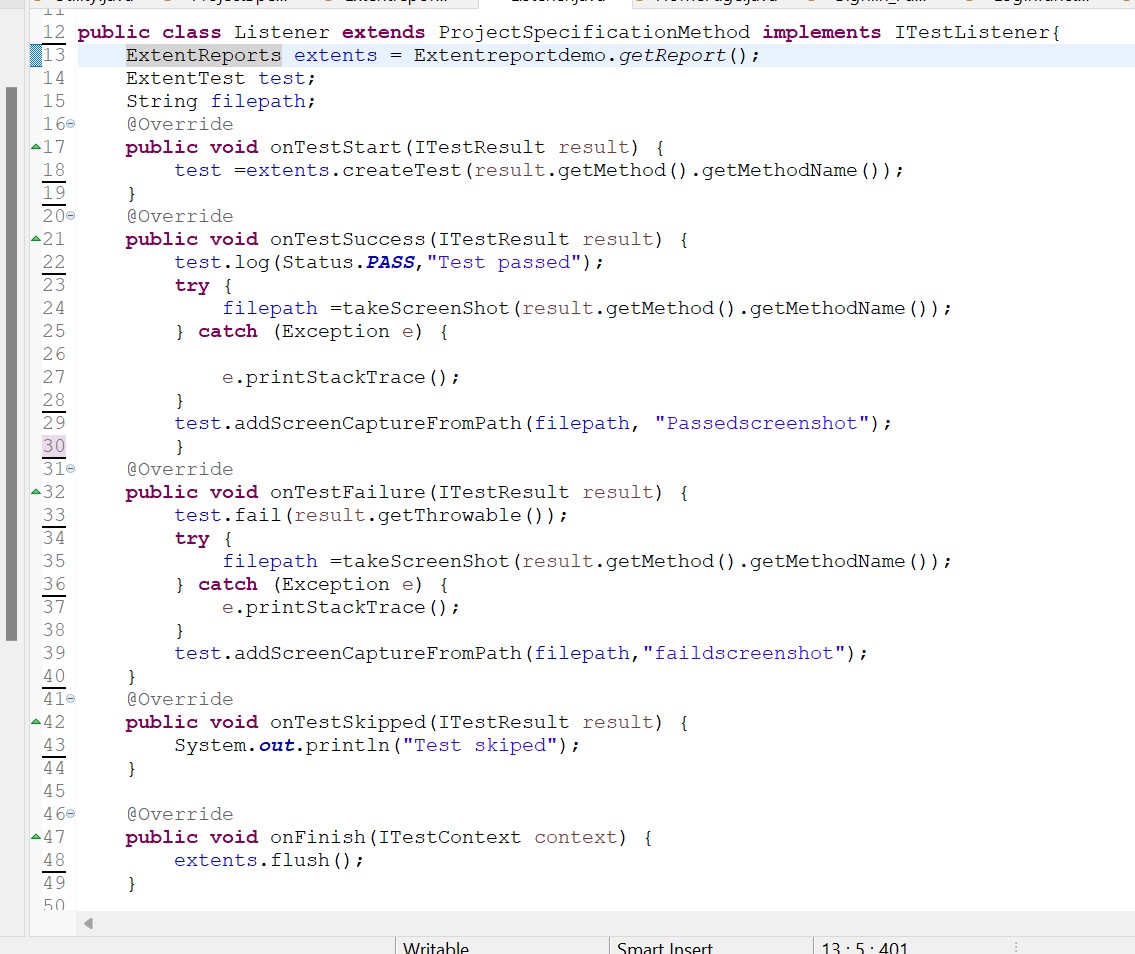
Org.base: Project specification class Org.pages: HomePage



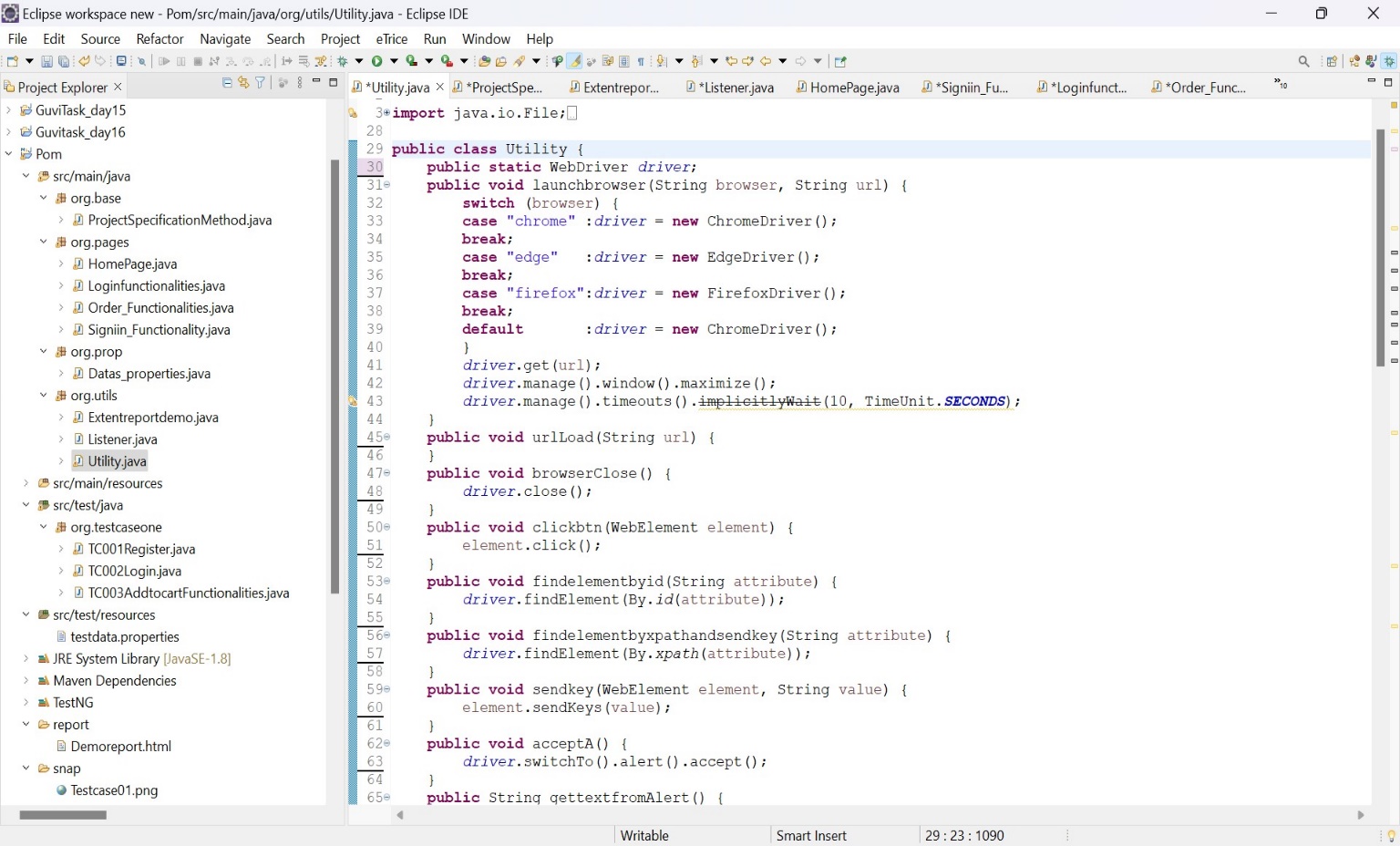
 Org.Pages:SigninFunctionalities Org.Pages:LoginFunctionalities

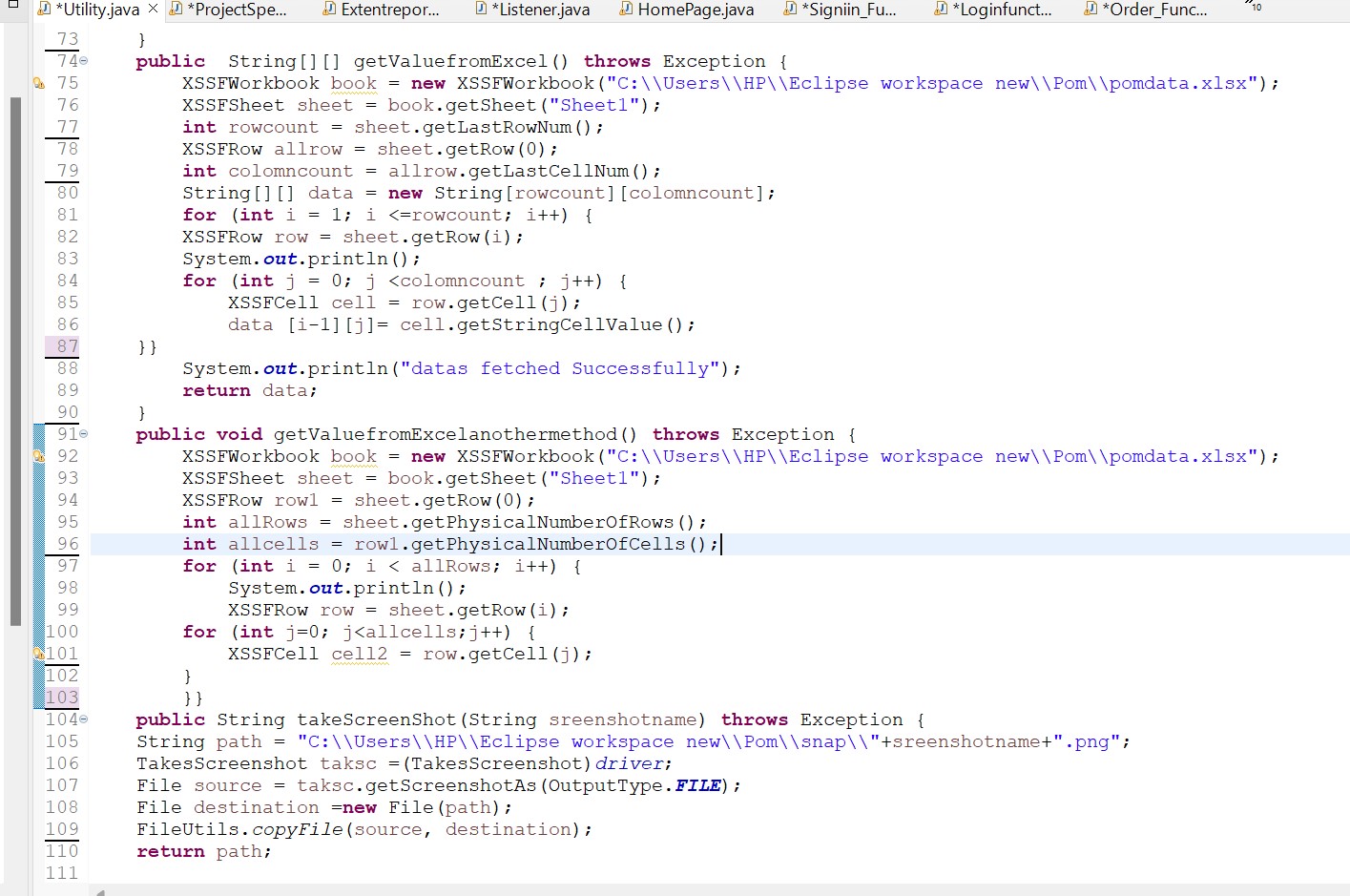
Org.Pages:AddtocartFunctionalities Org.utilis: ExtentReport



Org.utilis: Listener class

Org.Utils :Utility class





B) TesttNg

TestNG (Test Next Generation) is a widely used testing framework for Java that simplifies the testing process and makes it more powerful and easier to implement.

**Annotations**: TestNG uses annotations to define the test methods. Annotations like @Test, @BeforeSuite, @AfterSuite, @BeforeTest, @AfterTest, @BeforeClass, @AfterClass, @BeforeMethod, and @AfterMethod are used to mark the methods that should be executed as part of the test suite lifecycle.

**Test Methods**: Test methods are marked with the @Test annotation. These methods represent individual test cases.

**Groups**: TestNG allows you to categorize your tests using groups. You can include or exclude certain groups when running tests, which provides a way to organize and selectively execute tests.

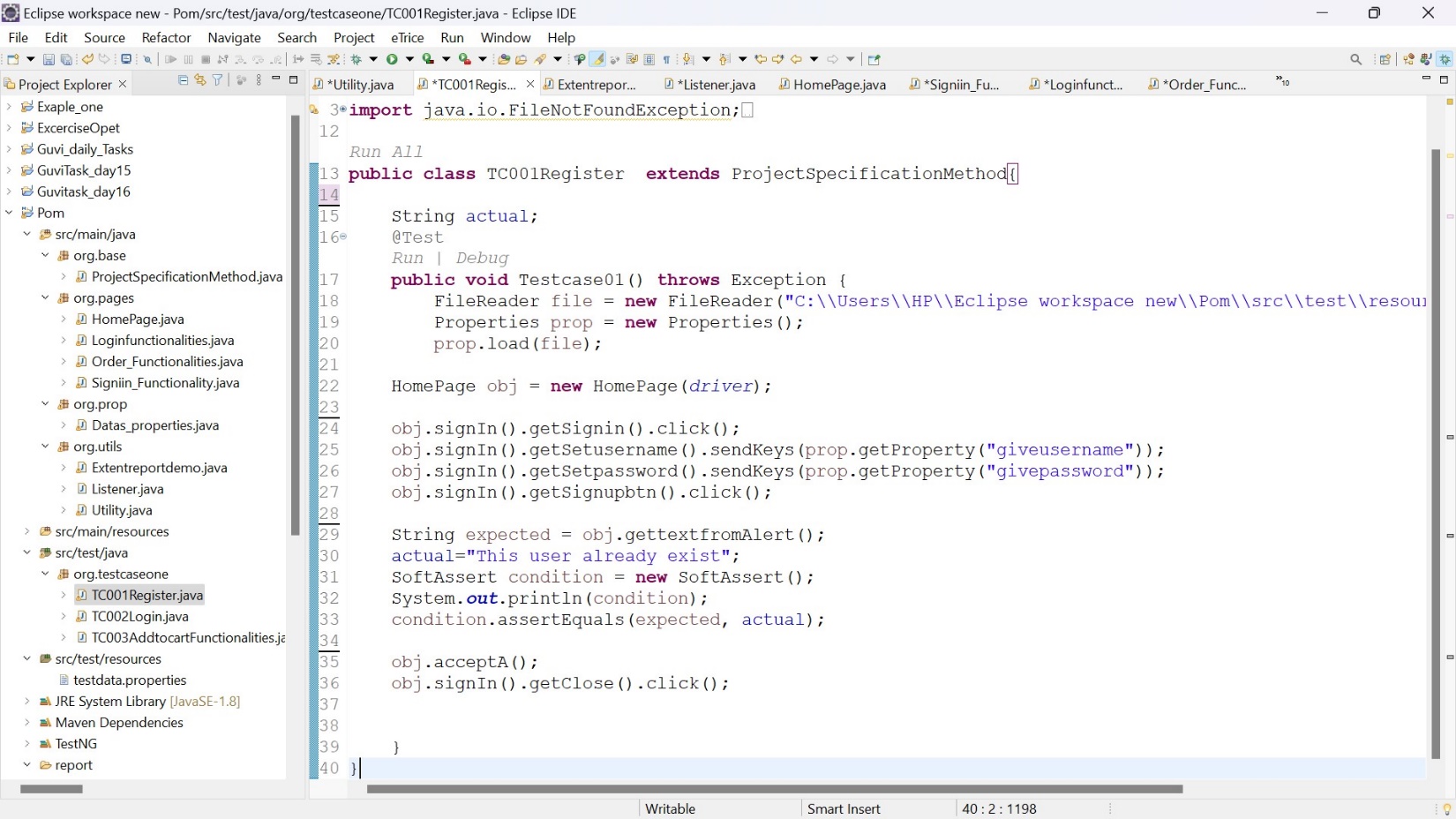
**Parameters**: TestNG supports parameterized testing, allowing you to run the same test method with different sets of data.

**Listeners**: TestNG provides the ability to use listeners to customize the test execution behavior. You can implement listeners to perform actions before or after test methods, suites, and other test-related events.

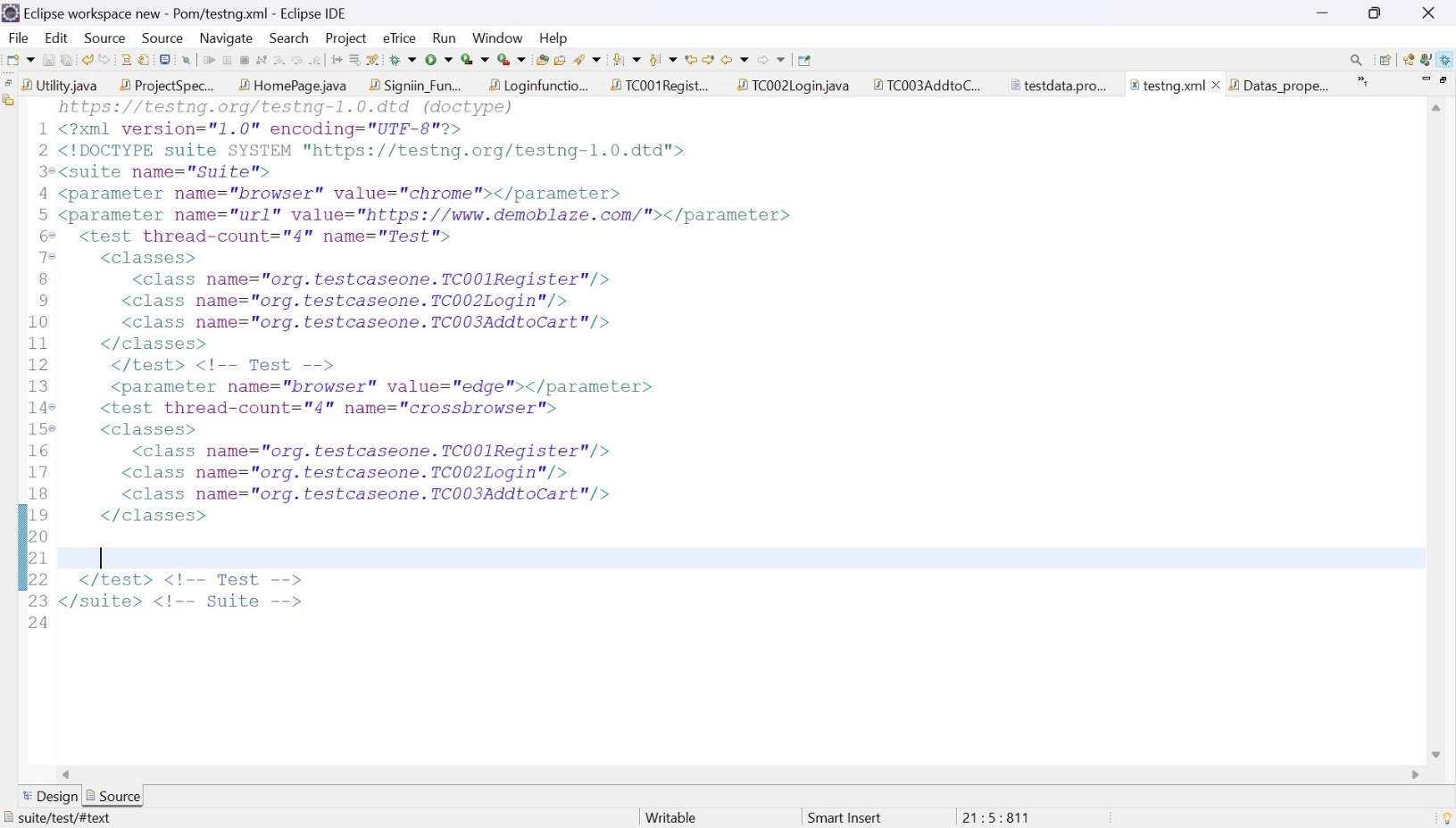
**Parallel Execution**: TestNG supports parallel test execution, which can significantly reduce the overall test execution time.

**Data Providers:** TestNG allows you to use data providers to supply data to test methods, enabling data-driven testing.

Use assertions to verify that the expected results are achieved.



* Here I have used assertion
* When I have try to set username and password in signin functionalities, the user name is already exit means it will not create the account and terminate the code so I have use assertion in test ng
* For testing purpose I have declare soft assertion (Instead of stopping the execution of the test at the first assertion failure (as is the case with traditional or "hard" assertions), a soft assertion records the failures and allows the test to continue running.) so it allows moving the next step to test.

SuitLevel Execution :

* For suite level execution first we have to select all class and click right click and select option convert to TestNG and once finish the testng.xml file will create
* So we can execute all the classes one by one in testng.xml class

Cross Browser testing

* Cross-browser testing is the process of testing a web application across different web browsers to ensure its compatibility and functionality are consistent across various browsers.
* TestNG provides a way to perform cross-browser testing by configuring test methods to run on different browsers using the **@Parameters** annotation

<parameter name=*"browser"* value=*"chrome"*></parameter>

<parameter name=*"url"* value=*"https://www.demoblaze.com/"*></parameter>

<parameter name=*"browser"* value=*"edge"*></parameter>

Listener Tag

listeners are used to customize and control the behavior of the test execution. The **<listeners>** tag in the testng.xml configuration file allows you to specify which listeners should be used during the execution of the test suite.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<listeners>

<listener class-name=*"org.utils.Listener"*></listener>

</listeners>

<parameter name=*"browser"* value=*"chrome"*></parameter>

<parameter name=*"url"* value=*"https://www.demoblaze.com/"*></parameter>

<test thread-count=*"5"* name=*"Test"*>

<classes>

<class name=*"org.testcaseone.TC002Login"*/>

</classes>

</test> <!-- Test -->

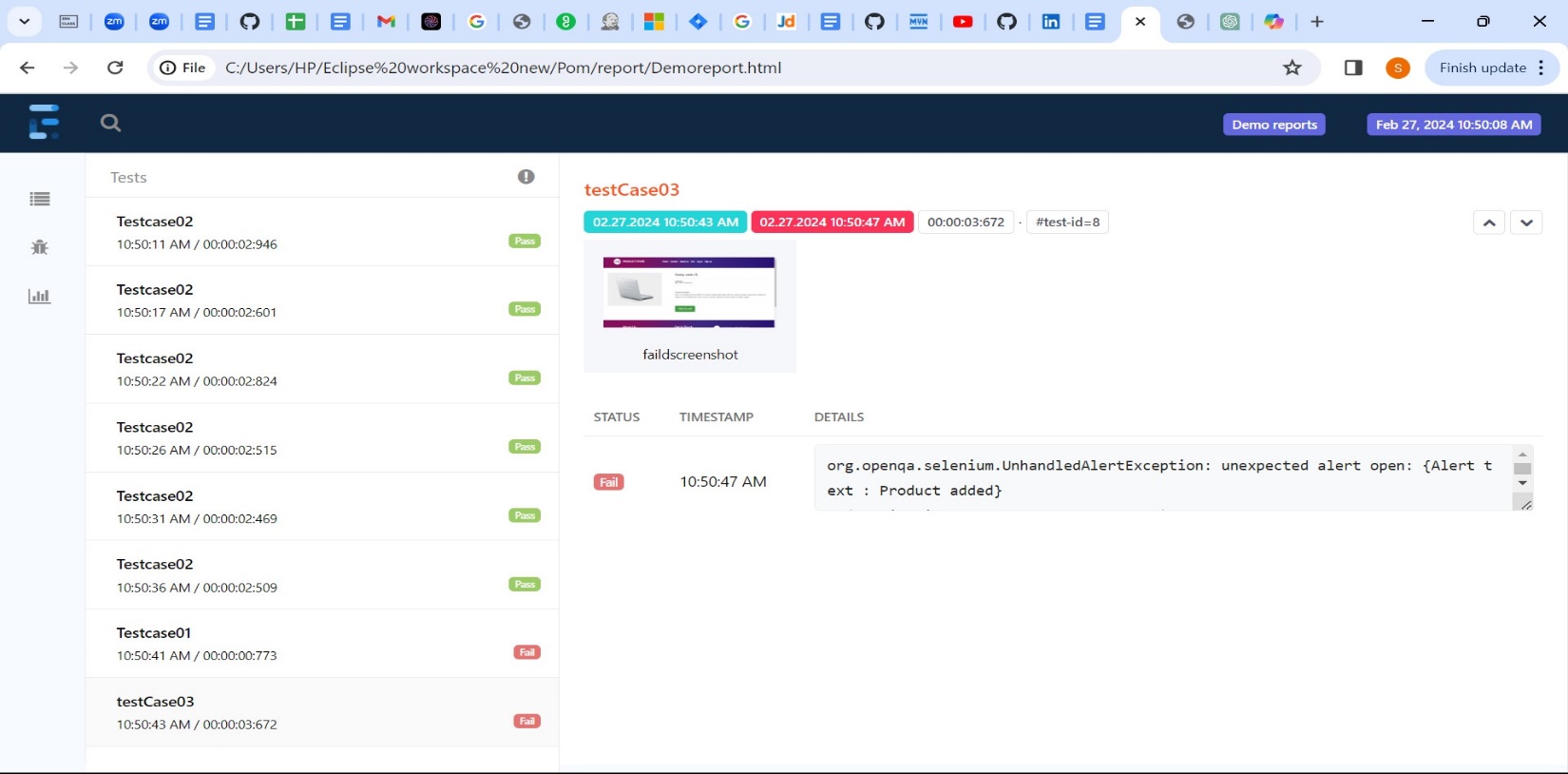
</suite> <!-- Suite -->

Extent report

Extent Reports is an open-source reporting library for TestNG (and other testing frameworks) that provides enhanced HTML reports for better visualization of test results. It offers detailed information about test executions, including pass/fail status, logs, screenshots, and more. Extent Reports can be integrated with TestNG to generate visually appealing and informative HTML reports.

**Project Report url:** [file:///C:/Users/HP/Eclipse%20workspace%20new/Pom/report/Demoreport.html](C://Users/HP/Eclipse%20workspace%20new/Pom/report/Demoreport.html)

Output : 1 Passed testcase



OutPut : 2 Failed Testcase

