SMART VERIFY

Test Automation Documentation

**Contents**

1. Test Automation Framework
2. Hybrid Test Automation
3. Pre-requisites
4. Automation Project Set Up
5. Framework Folder Structure
6. Framework Architecture
7. Pictorial Representation of the Framework Flow
8. Framework Flow Explanation
9. Test Data Explanation
10. Steps to Execute the Test Scripts.
11. Result Analysis

**Test Automation Framework**

 A testing framework is a set of guidelines or rules used for creating and designing test cases. A framework is comprised of a combination of practices and tools that are designed to help QA professionals test more efficiently.

**Hybrid Automation Framework**

Hybrid Test automation framework is the combination of two or more frameworks. Its uses strengths and benefits of other frameworks for the particular test environment. Test automation frameworks boost productivity with standardization.

**Pre-requisites**

* JDK 1.8
* TestNG
* Eclipse IDE
* Must have a Git Hub account
* Driver executables should be updated manually as per the current version of browser
* Data Base reset must be done before execution

**Code Location**

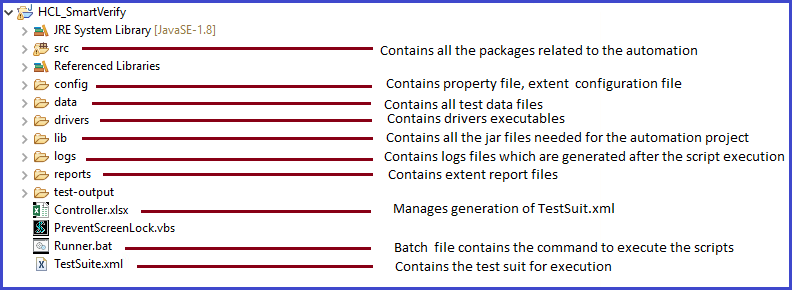
https://github.com/TYSS-HCL/SmartVerify.git

**Automation Setup**

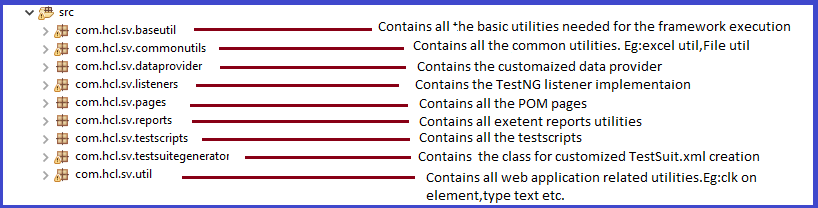
1. Open eclipse.
2. Create a workspace where you want to store the project.
3. Click on go to workspace symbol on the eclipse welcome screen.
4. Click on Window-> Show View-> Project Explorer.
5. Right Click in package Explorer, import->import->Projects from git->Clone URI->Enter git URI Provide username and password->select the main branch->next-> finish.

**Framework Folder Structure**

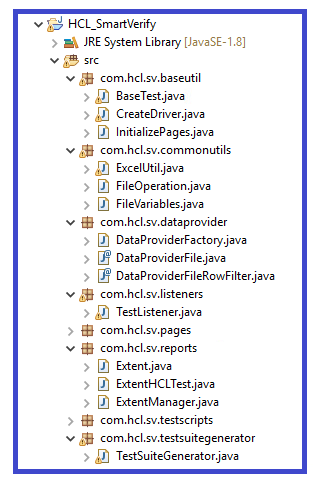
The following figure describes the folder structure with the description.

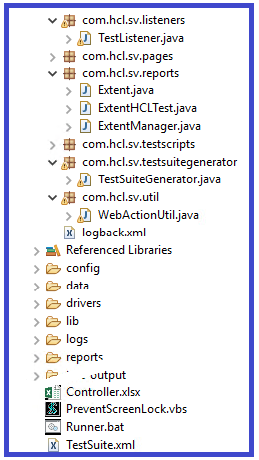


Below pictures represents the package structure of the automation framework.



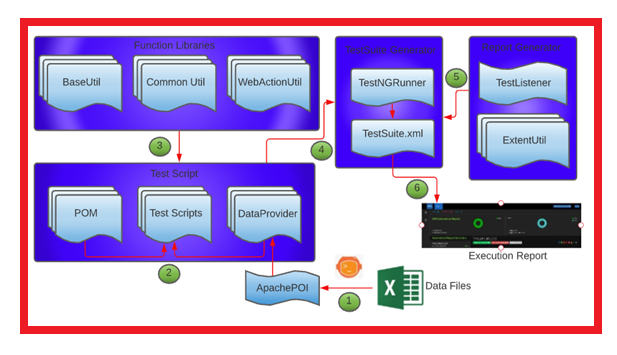
Below picture shows the different java classes under the packages which have the implementation of all the generic as well as the specific methods to be used to test the web application.





**Framework Architecture**

Below figure depicts framework architecture along with the components used in project.



**Base Util**: Contains Base Test, Create Driver and InitializePage.java. Base Test contains all the pre- and post-condition implementation, such as creating the folder structure for the reports, Launching the browser and entering the URL, close the browser. Create driver will be used to launch the specified browser. InitializePage.java has the object creation of all the POM pages.

**CommonUtil**: Contains Excel Util, File Variables, and File Operations classes. Excel util has the implementations of all the excel functions using apache poi, fillo. File Operations and Variables contain the implementations of the File handling needed in the application.

**WebActionUtil**: Contains all the implementations of the actions to be performed on the browser.

**POM**: Contains all the pages of the application. All the pages are created using Page Object Model Approach. These pages contain all the reusable methods.

**Test Scripts**: The Test cases are converted into the scripts using the reusable methods from the POM pages, data providers.

**Data Providers**: The customized data providers are used to fetch the data from the excel file.

**TestListeners**: Contains the implementations for methods from the ITestListener with extent reports implementations.

**ExtentUtil**: Contains the implementation of the extent reporting.

**TestNGRunner**: This class is used to generate TestNG.xml (anyname.xml) dynamically which is used to drive all the TestNG classes. This reads Controller.xlsx file which contains package name in a column and class Name in another column named TestcaseId. Based on the status 'Yes'/'No’, the TestNG.xml will contain only those classes where status is 'Yes'.

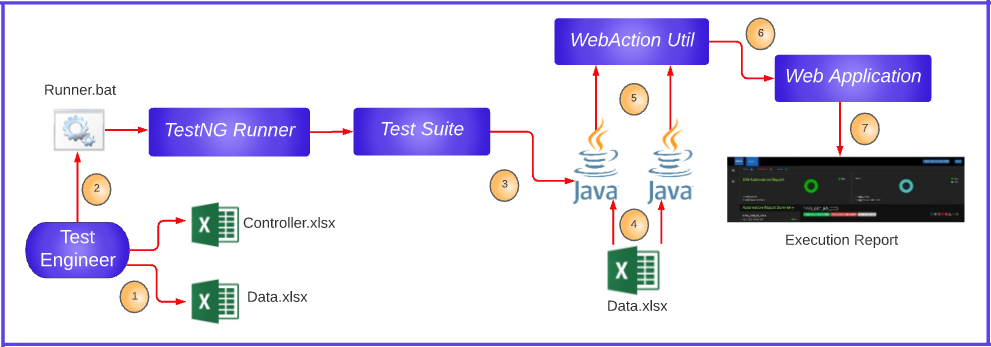
**Config**: This folder contains extent-config xml file, config properties.

**Drivers**: This folder contains all the driver executable files for launching the browsers.

**Lib**: This folder contains all the jar files required for this project.

**Data**: This folder contains all the test data files required for this project.

**Pictorial Representation of the Framework**



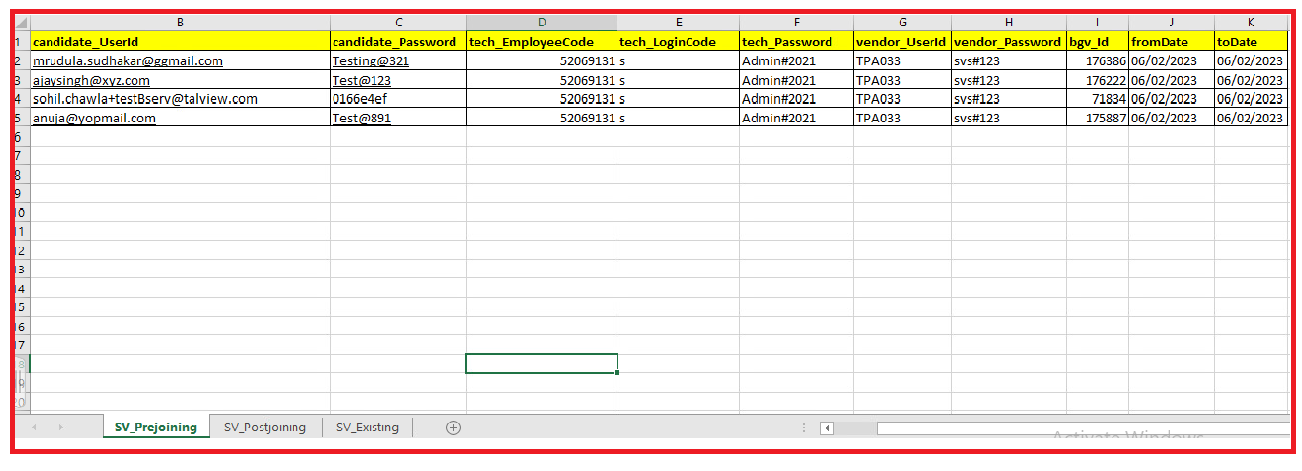
**Framework Flow Execution**

* **Step 1**: The Test Engineer modifies the controller.xlsx and data.xlsx file.
* **Step 2**: The Test engineer triggers the Runner.bat file. The Runner.bat triggers the execution of the whole framework. The TestNG runner creates the TestSuite.xml depending on the Controller.xlsx. The selection of the test scripts for the execution and mode of the test execution can be controlled by the values in the controller.xlsx file
* **Step 3**: The Testsuite.xml executes the test scripts.
* **Step 4**: The test script fetches the data from the Data.xlsx file with the use of a data provider.
* **Step 5**: Test scripts trigger the base test where actions like creating the folder structure for extent reports, launching the applications, creating the node for the test in the extent reports according to the test class. Also triggers the Web Action util.
* **Step 6**: The selenium script performs action on the web application under test using the common utilities and action utilities.
* **Step 7**: The results of the execution are recorded in the extent report.

**Test Data Preparation**

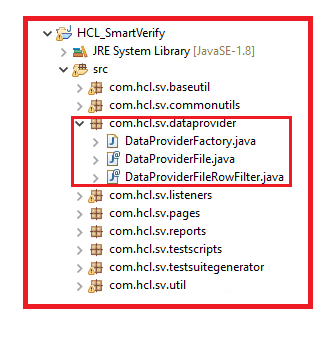
**Data preparation based on test script:** (EG: SV\_PREJ\_001)

1. The data is updated in different sheets according to the Flow name.

****

**Data Fetching Explanation:**

Customized data provider code is written in DataProviderFactory class.



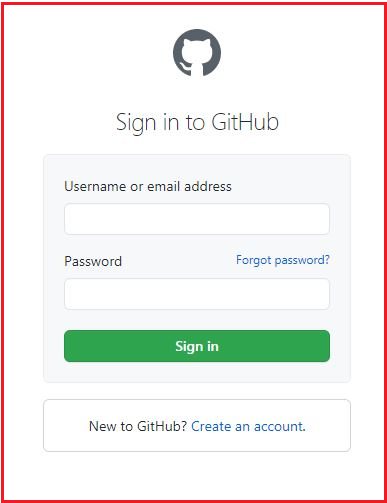
**How to Import Code From Git Hub**

**Method 1: Import the code in Eclipse by using Projects from Git:**

1. Open the given URL <https://github.com/> and Click on sign in.



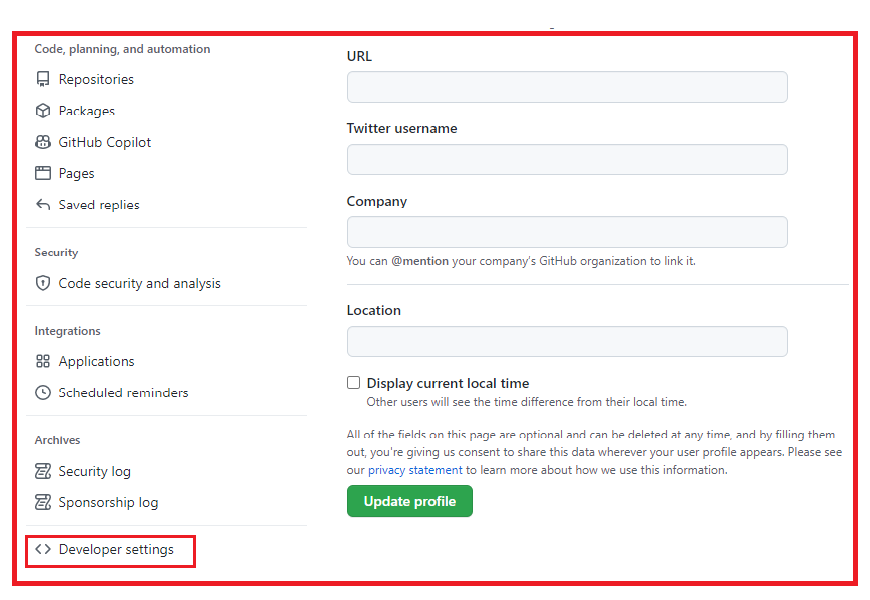
2. Enter your credentials and click on sign in.



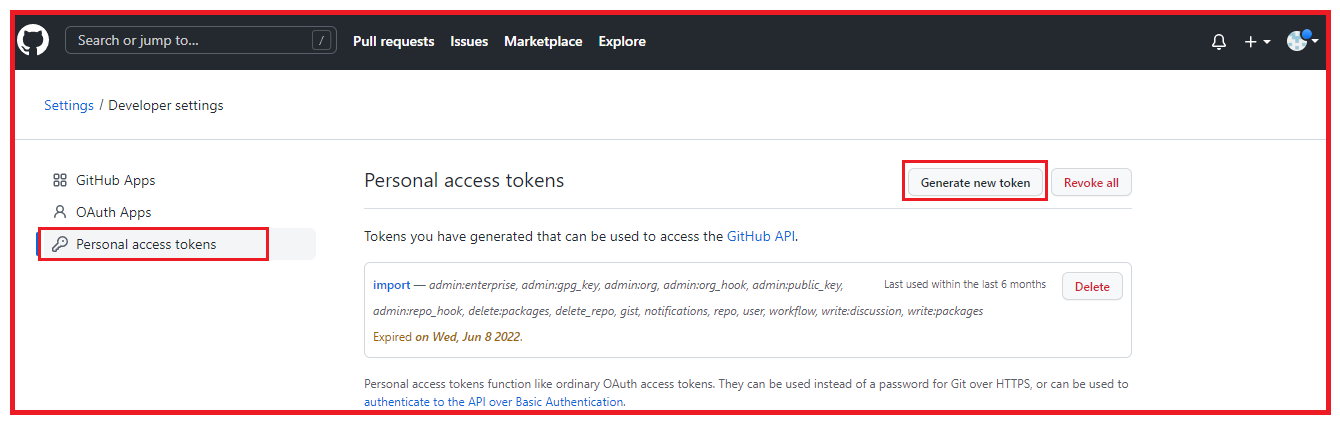
* Create the Token in git hub by following below steps:

3. Click on the My Profile dropdown & select settings.

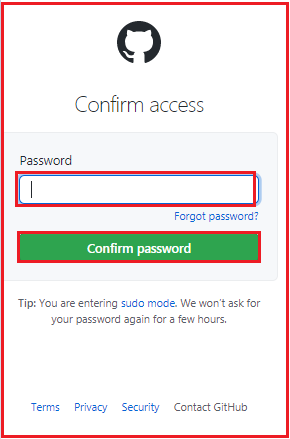


4. Click on Developer settings

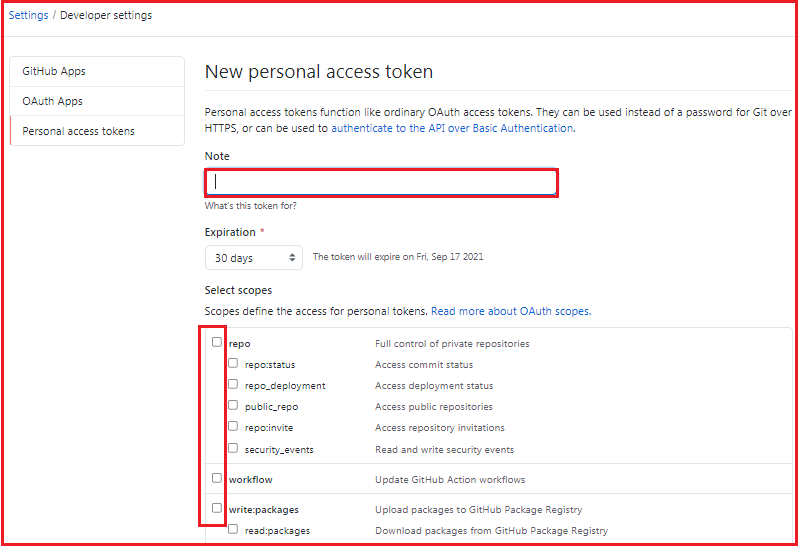
5. In developer settings, select Personal access token & then click on Generate new token.



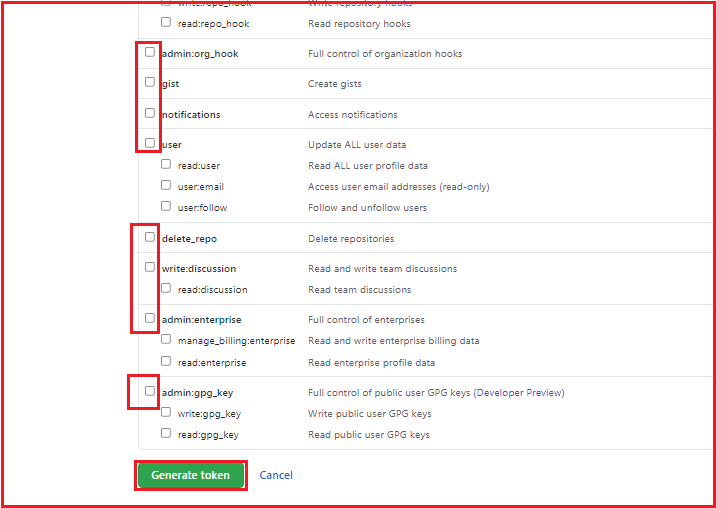
6. Enter the GitHub account password.



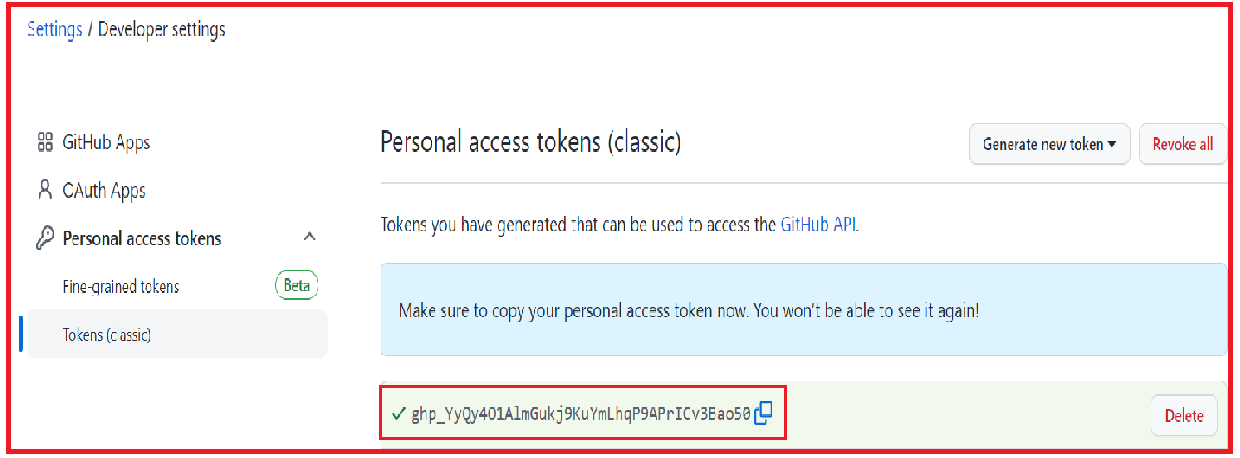
7. Provide any name in **Note** text field & select all the checkboxes.



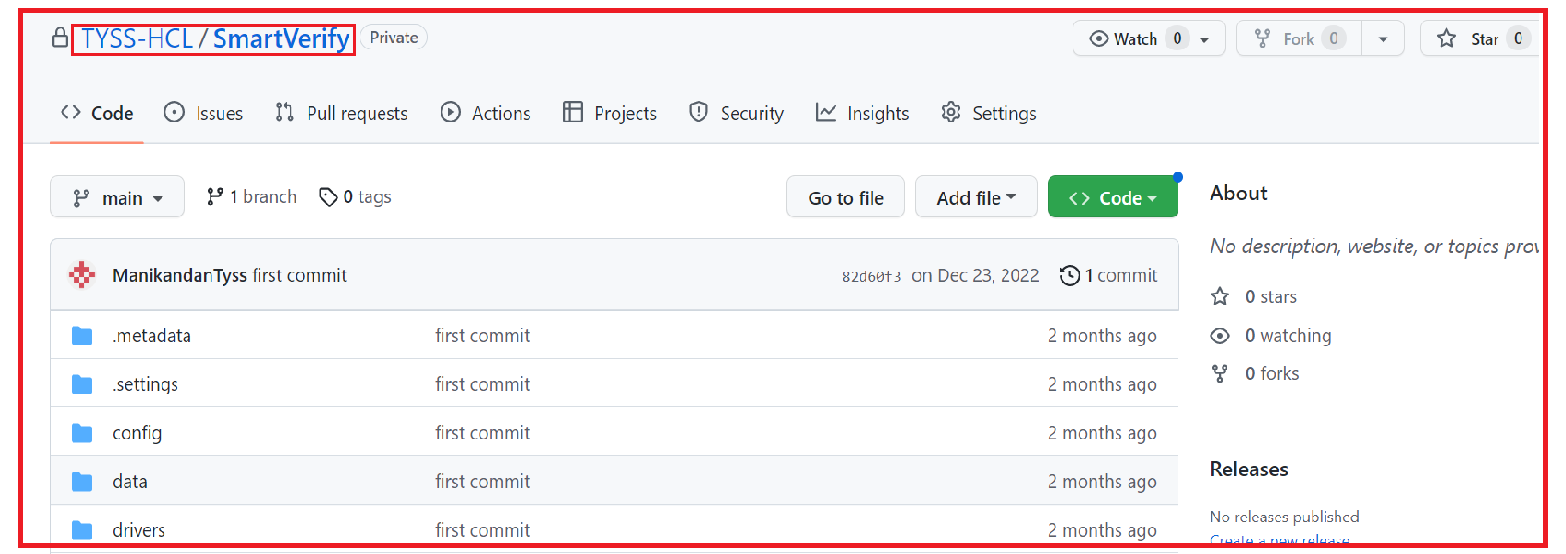
8. Click on Generate token.



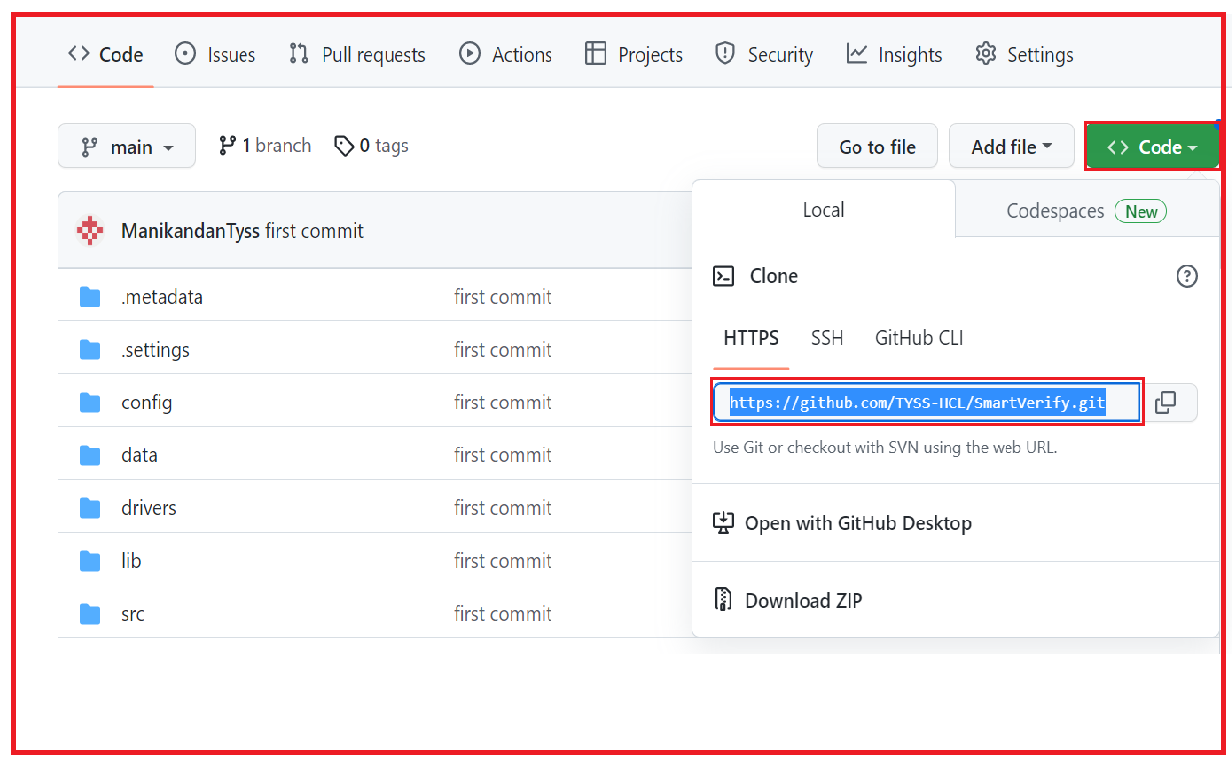
9. Copy & save the generated token for future reference.



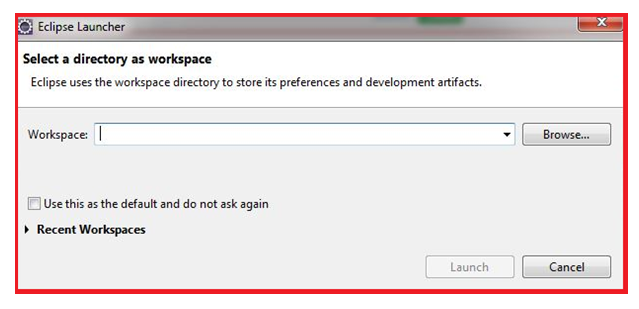
10. Project has been shared to the mentioned Git account by the Lead/Manager. Accept the invitation in email and you can see the project in your git account. Click on project.



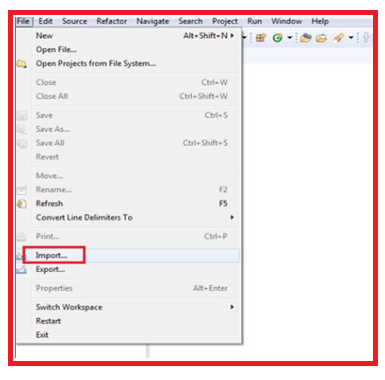
11. Click on code and copy the URL.



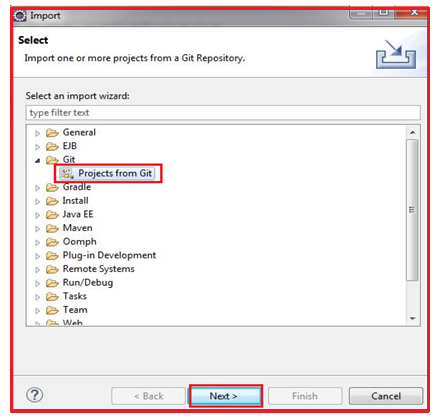
12. Open the eclipse and create a workspace (Create a folder in your system) and click on launch.



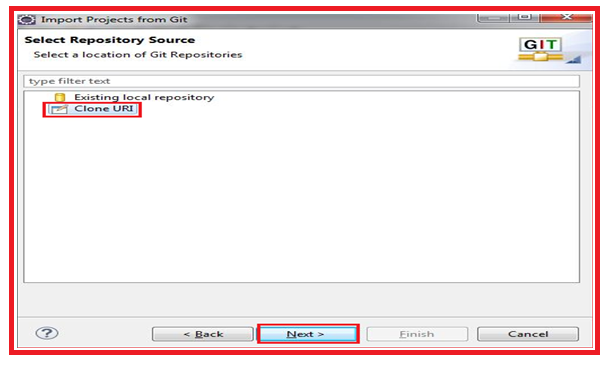
13. Click on file then click on import.



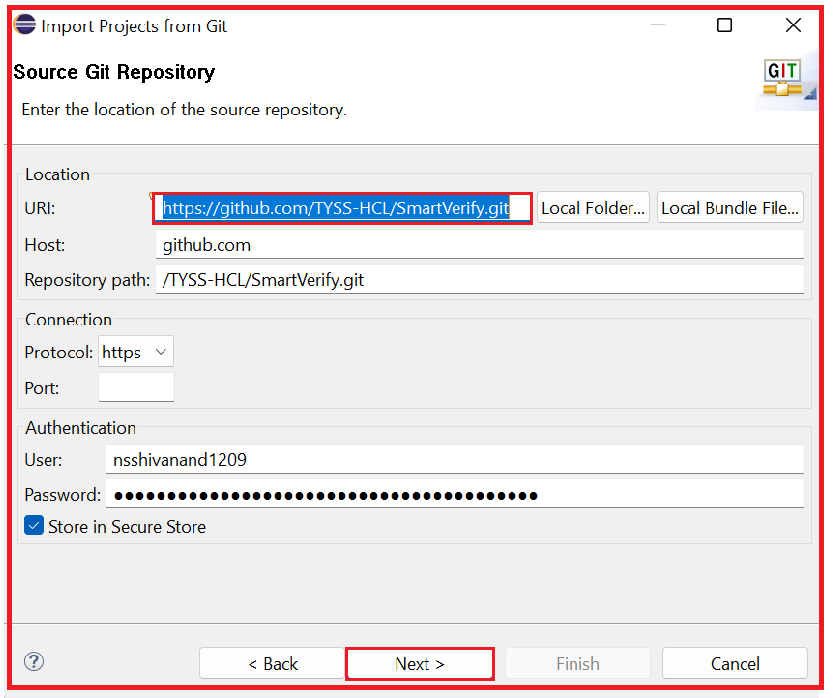
14. In Git option click on Projects from Git then click on next.



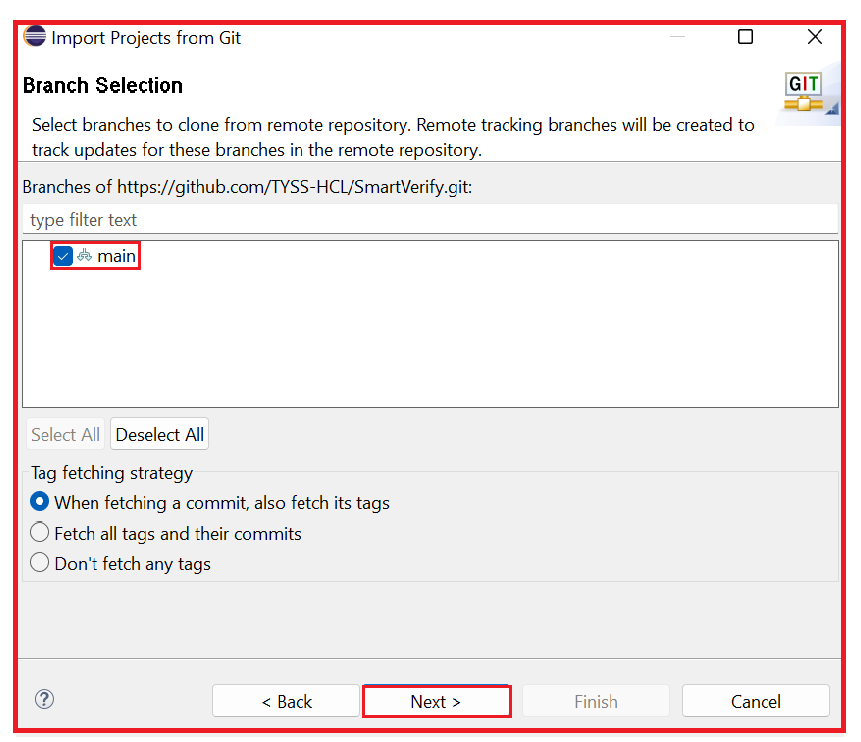
15. Click on Clone URI then click next.



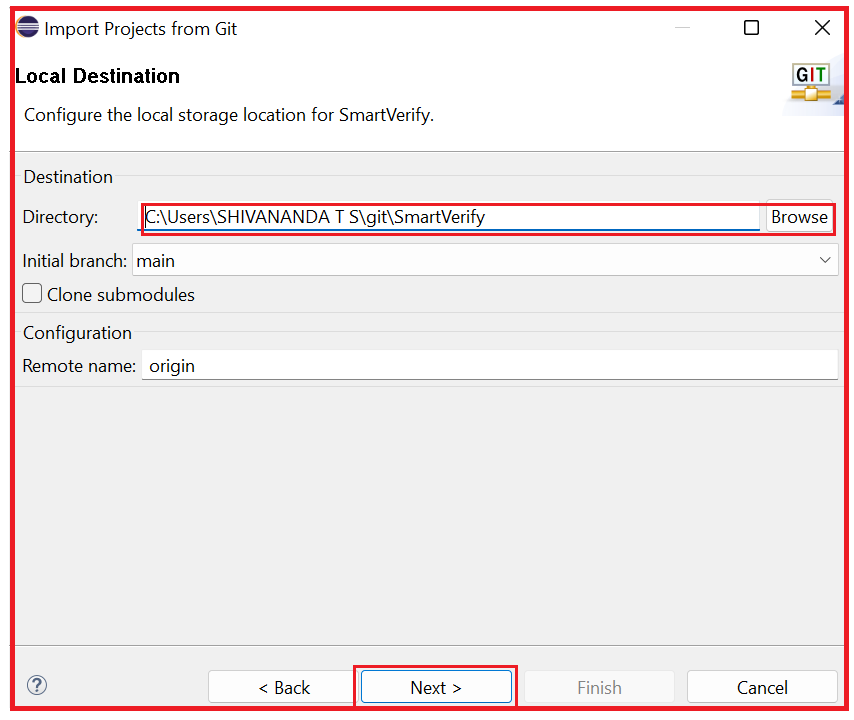
16. Paste the URL in URI section copied in step 11 and click on next button.



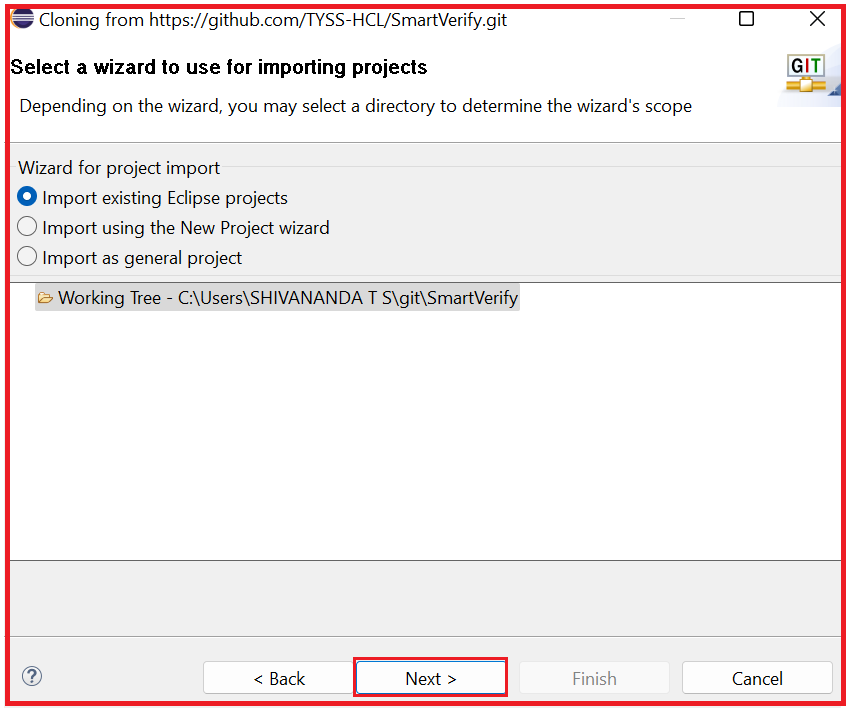
17. Select the master branch then click on next.



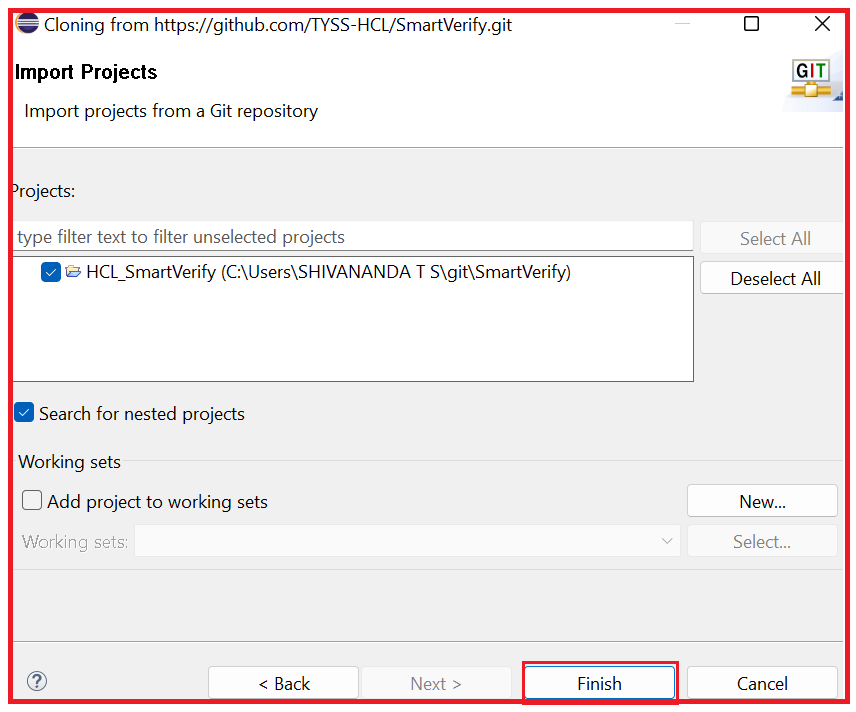
18. Select the directory for git then click on next.



19. Click on next.



20. Click on Finish.

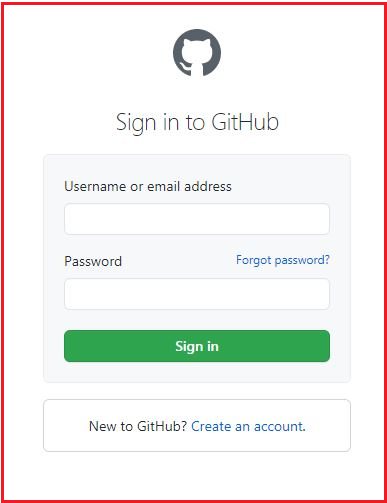


**Method 2: Import the code in Eclipse from downloaded ZIP file:**

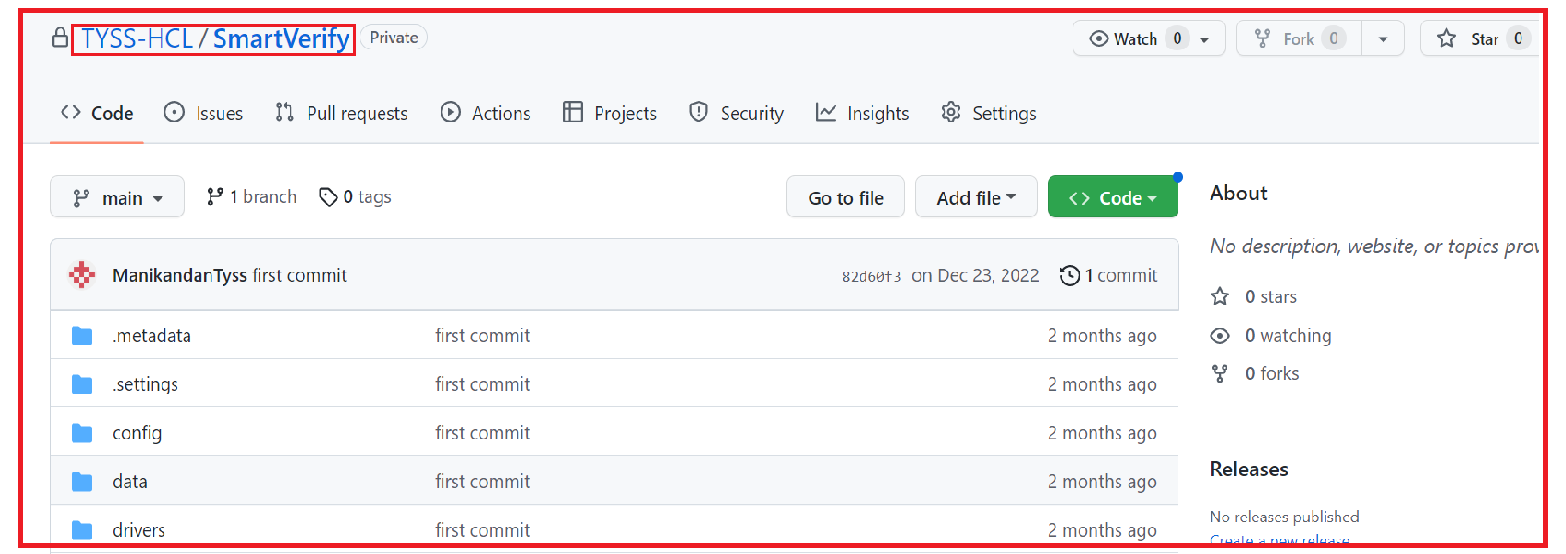
1. Open the given URL <https://github.com/>and Click on sign in.



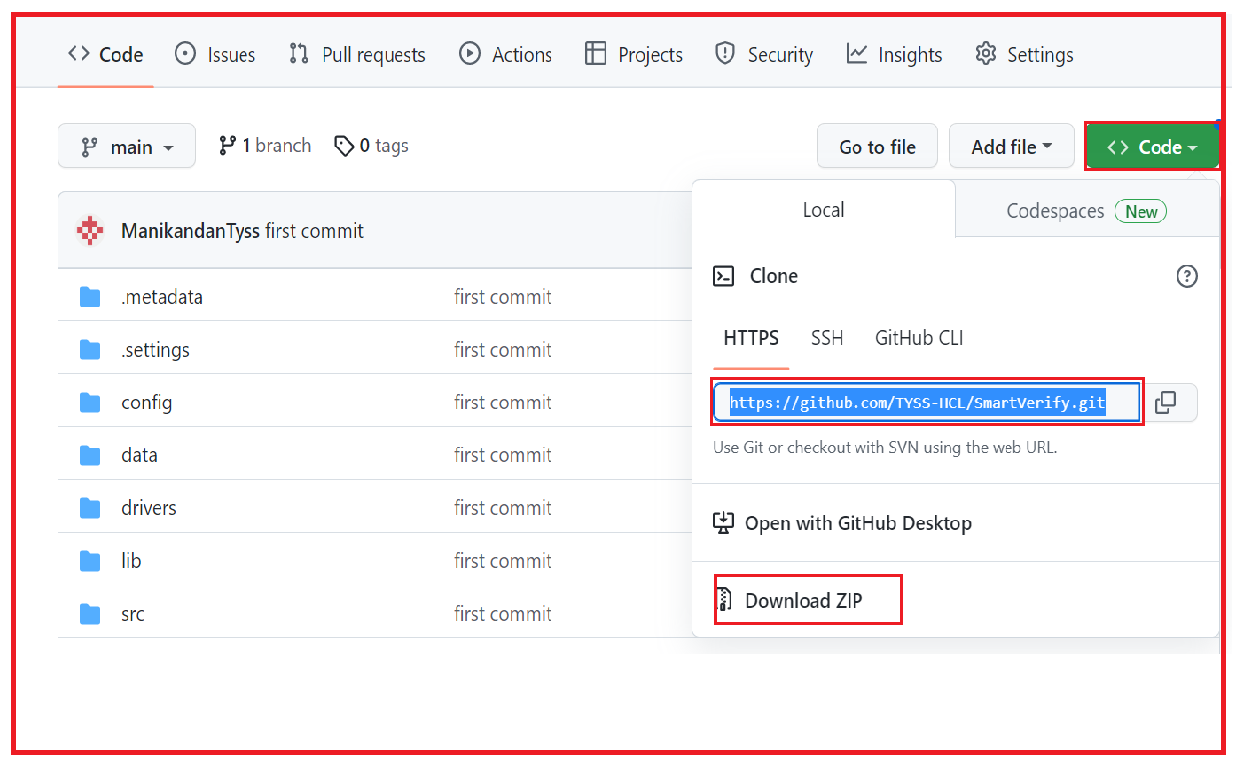
1. Enter your credentials and click on sign in.



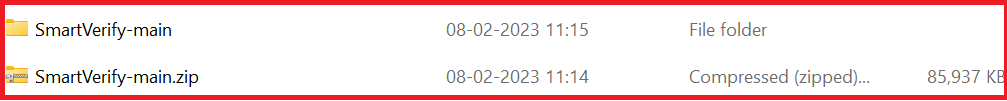
3. Project has been shared to the mentioned Git account by the Lead/Manager. Accept the invitation in email and you can see the project in your git account. Click on project.



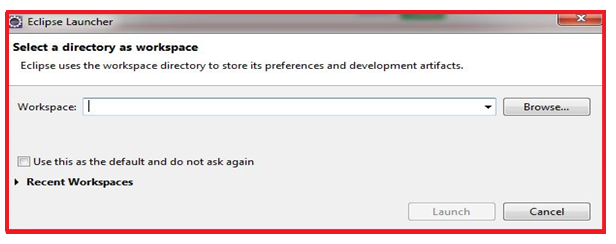
4.Click on code then click on Download Zip. Zip file get downloaded in your system.



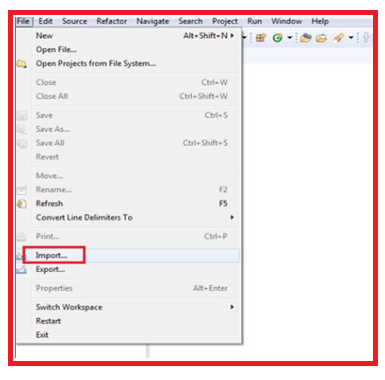
5. Unzip the file.



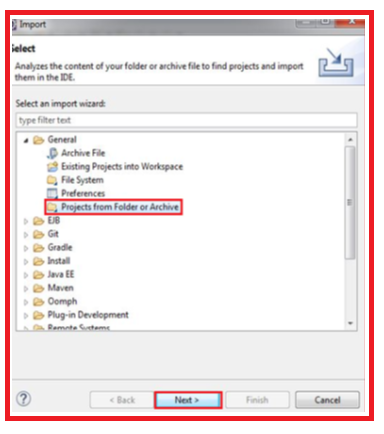
6. Open the eclipse and create a workspace (Create a folder in your system) and click on launch.



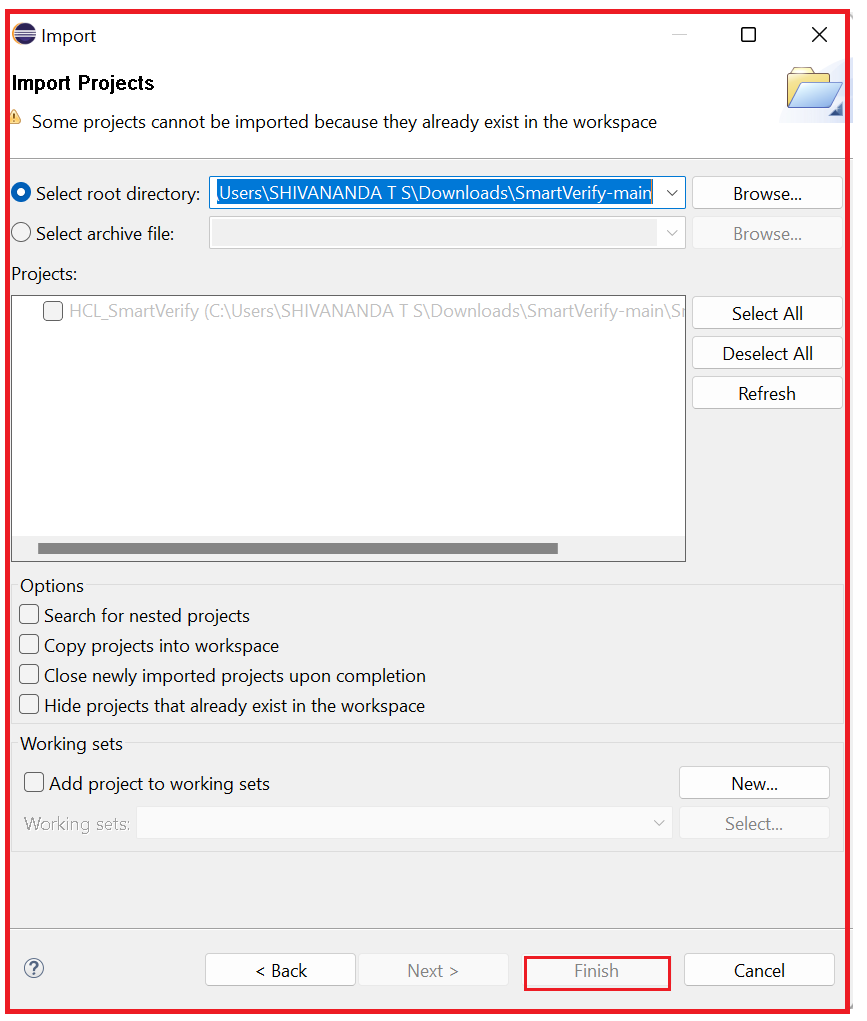
7. Click on file then click on import.



8. In General option click on Projects from Folder or Archive then click on next.



9. Select the downloaded unzipped folder through directory then click on finish.

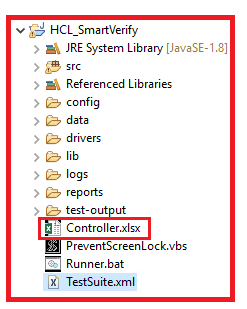


**Method 1: Using Eclipse IDE:**

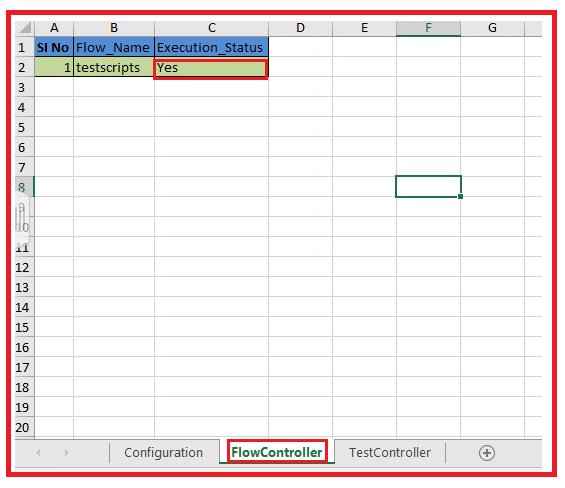
**Note:**

* **Data Base reset must be done before execution.**
* **Before starting the execution, username and password of VDI must be mentioned in the config file.**

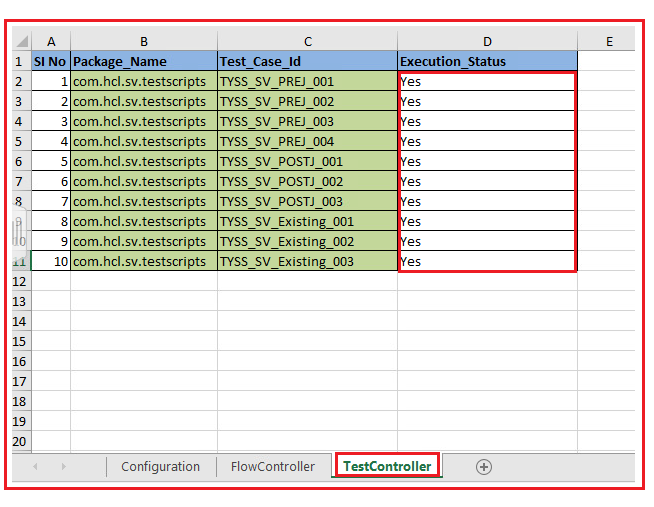
1.Expand the Project in the Eclipse and open the Controller.xlsx file.



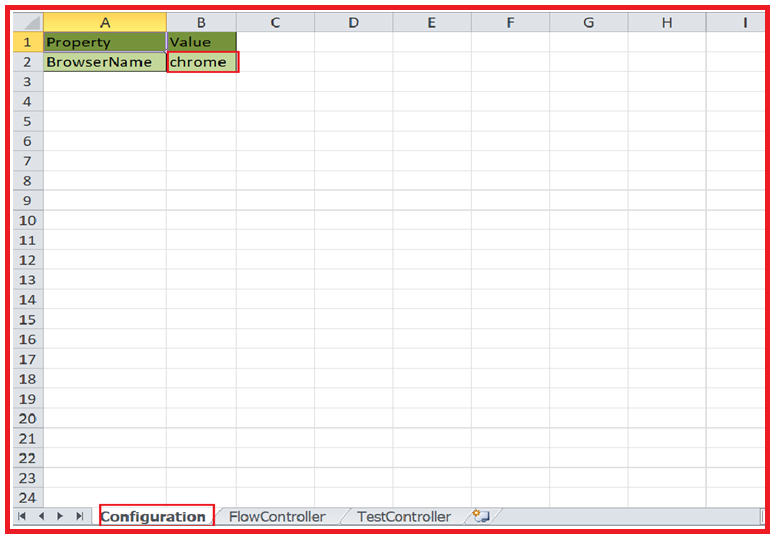
2.Enter the execution status as ‘YES’ for the Flow Name that needs to be executed in the Flow Controller sheet.



1. Go to the Test Controller sheet and change the execution status to “Yes/No” as per the requirement.

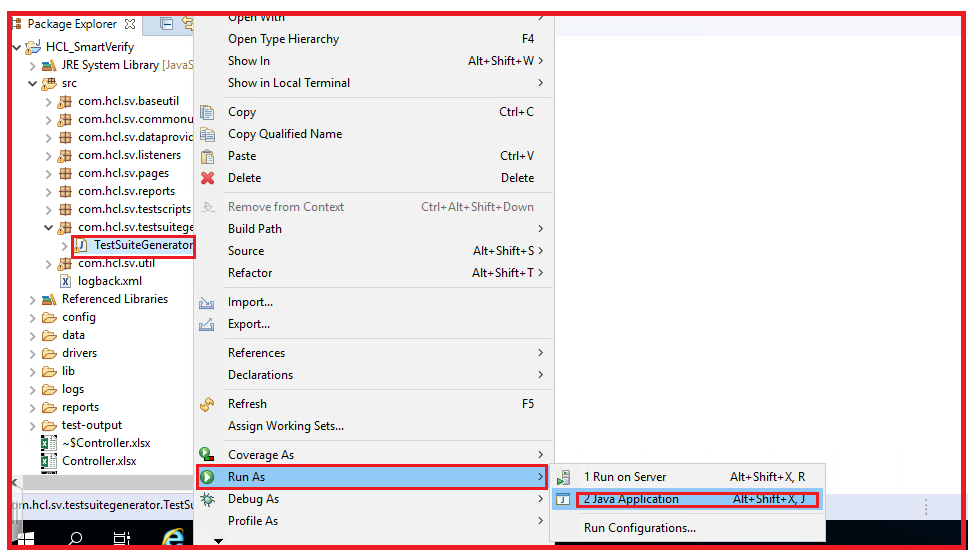


4. Go to the Configuration sheet and Mention the browser for execution.



5. Save the controller file.

1. Right click on the TestSuiteGenerator class and select run as java application.

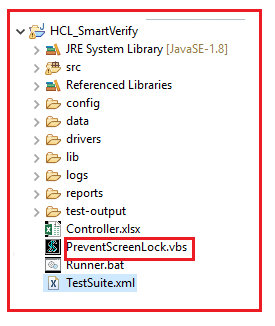


7. Refresh the project.

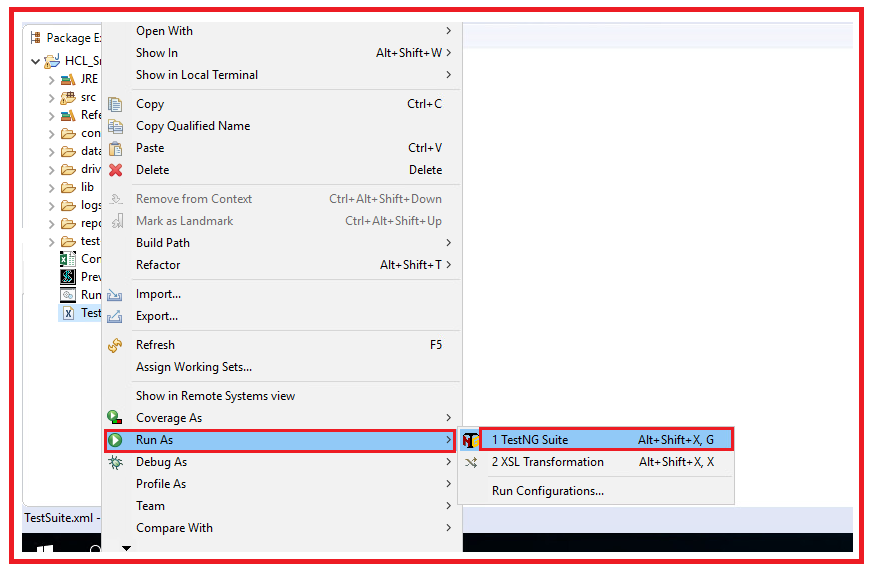
8. If you are using VDI then copy-paste PreventScreenLock.vbs in your VDI and double click on it.



9. Double click on PreventScreenLock.vbs file in eclipse.



10. Right click on TestSuite.xml, select run as TestNG suite.

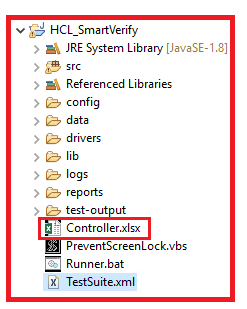


**Method 2: Using Runner.bat file**

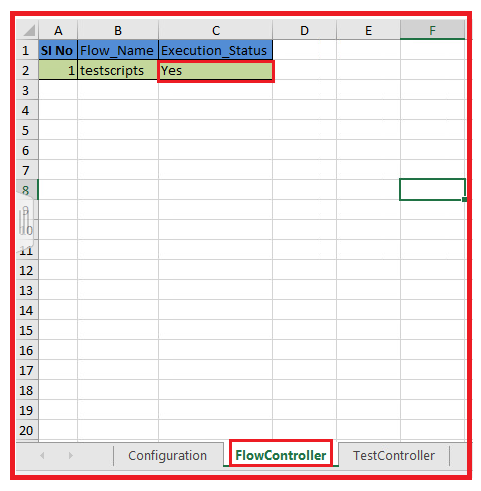
**Note:**

* **Data Base reset must be done before execution.**
* **Before starting the execution, username and password of VDI must be mentioned in the config file.**

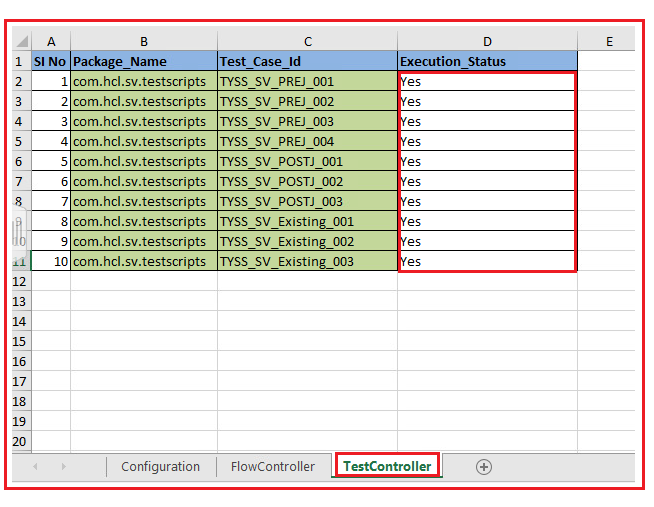
1. Expand the Project in the Eclipse IDE and open the Controller.xlsx file



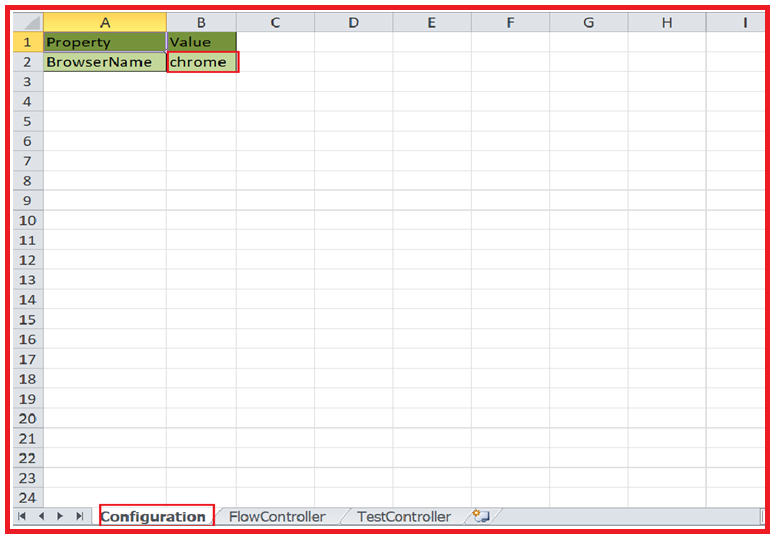
1. Enter the execution status as ‘YES’ for the Flow Name that needs to be executed in the Flow controller sheet.



3. Go to the Test controller sheet and change the execution status to “Yes/No” as per the requirement.



4. Go to the Configuration sheet and Mention the browser for execution.

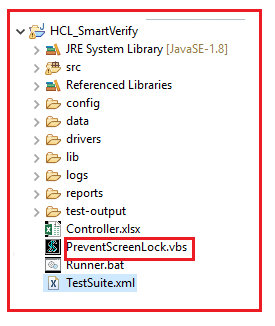


5. Save the controller file.

6. If you are using VDI then copy-paste PreventScreenLock.vbs in your VDI and double click on it.

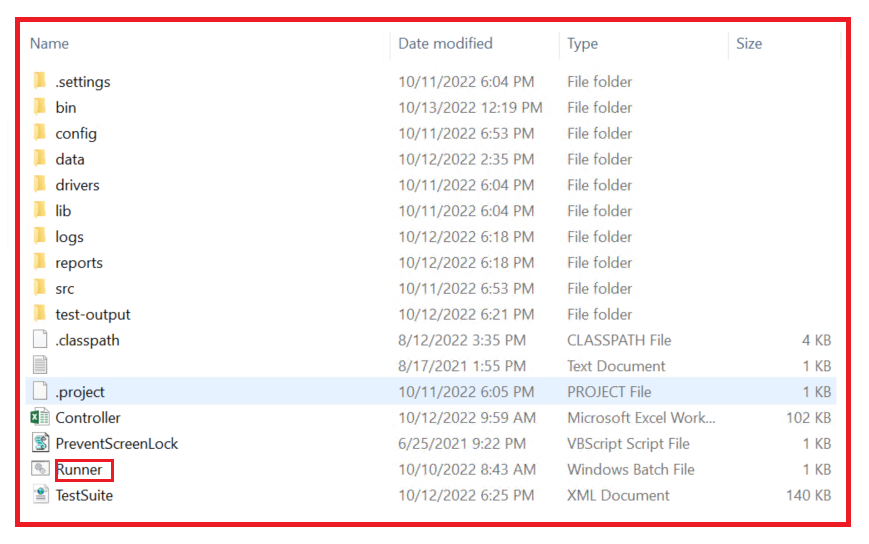


1. Double click on PreventScreenLock.vbs file in eclipse



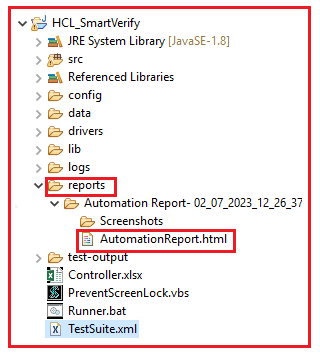
8. Double click on Runner.bat File in the source folder.

Runner.bat file internally execute the TestSuiteGenerator.java class which creates TestSuite.xml file and then execute this xml file.

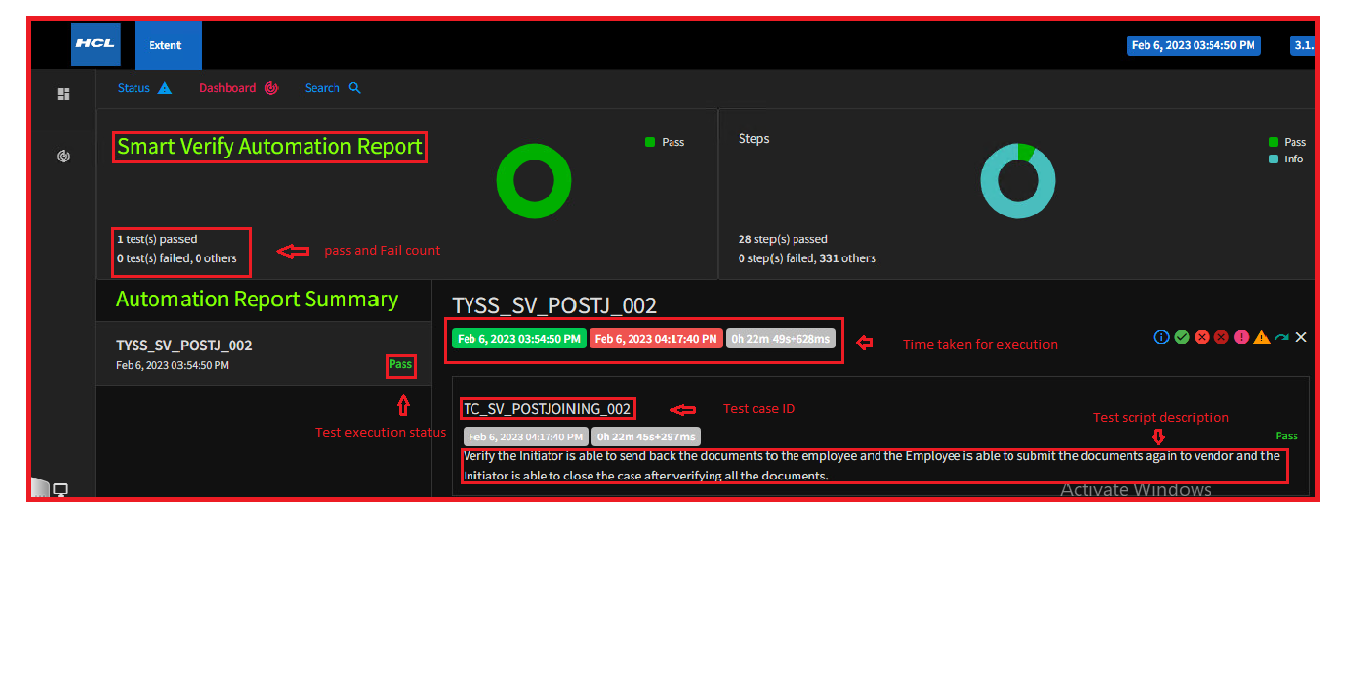
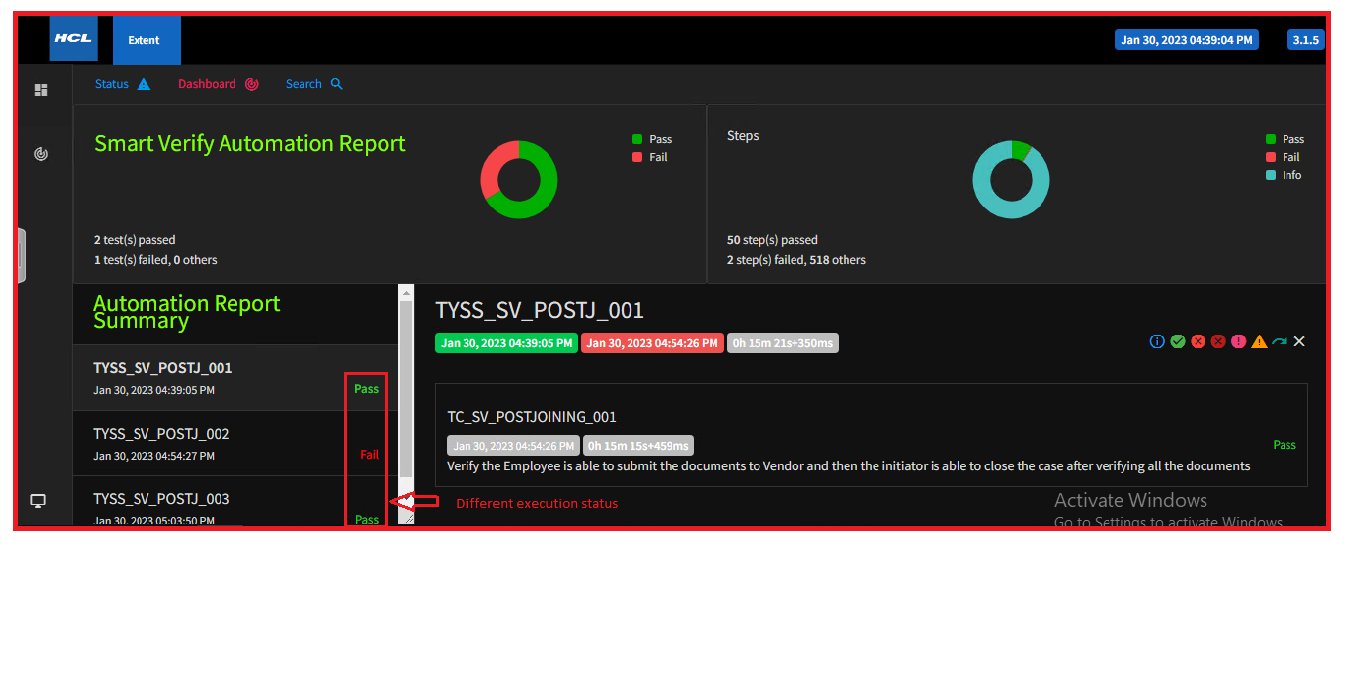


**Result analysis**

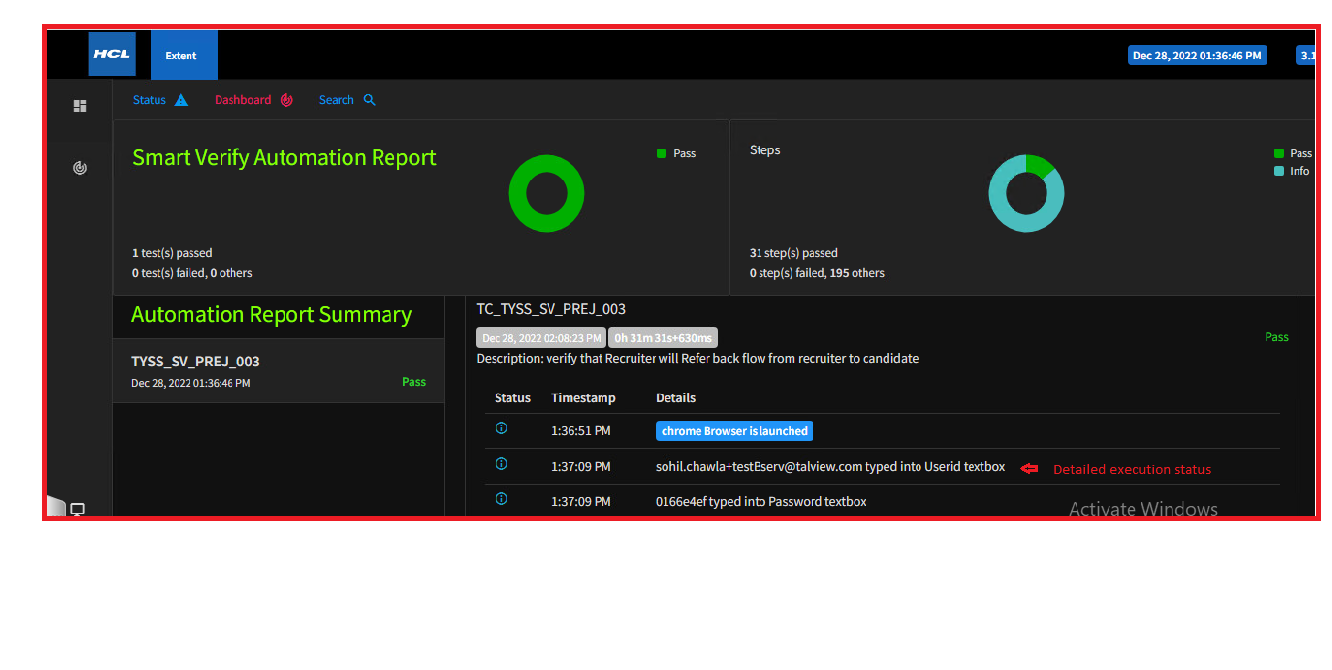
**Extent Reports**: After the execution the extent reports are saved in the **reports** folder in the project directory. Refresh the project directory and get the newly generated reports.



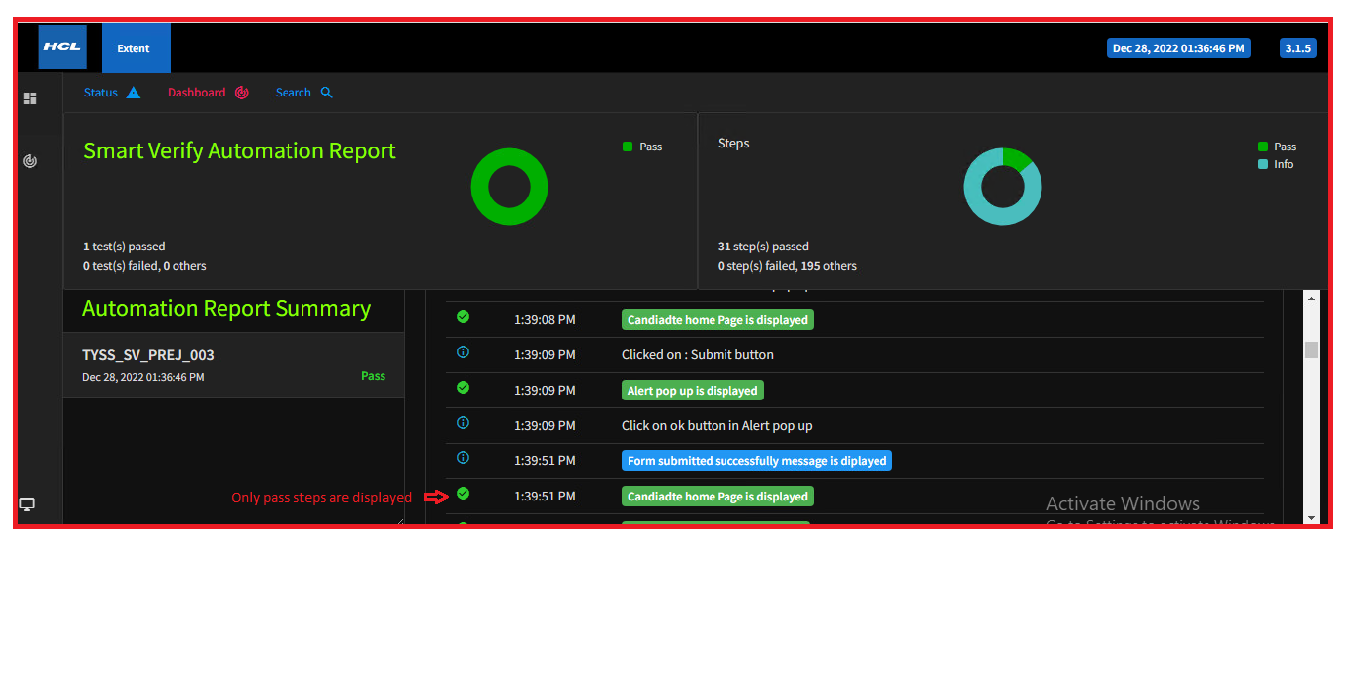
The below picture depicts the html report which is obtained in the report folder after the completion of the execution.

Below picture show s the different status obtained in the report after the completion of the execution.

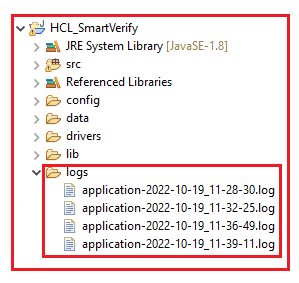
The below picture depicts the detailed execution steps. The major application validations are highlighted in blue; business validations are highlighted in green.



The following picture shows that user can apply the various filter & sort the detailed steps.



**logback.xml**: After the execution the logback reports are saved in the **logs** folder in the project directory.

****

The logs of the execution are updated in the logs file with the Timestamp as shown in below picture.

