**SSD Selector Tool Automation Testing Framework Details**

* I Have implemented the SSD Selector Tool, Automation testing using **Hybrid Driven Frame Work**.
* Its combination of **POM (Page Object Model)** and **TestNG** Framework.
* For Reports Analysis, I have used **Extent Report** for Graphical representational,
* Build Tool, I have used **Maven**
* Integration Tool, I have used **Jenkins**.

Page Object Design

Interface type

**SATAVRO, SATA, Value SAS, SAS, NVMe**

**Next**

Server Type Page, Server Model, Capacity Select

**Server Type, Model, Capacity Value**

**Next**

Workload Page VRO, RI, MU, WI

**VRO, RI,MU,WI**

**Next**

Home Page

I Know What I Need

Help Me Choose

Result /Recommendation Page

**SSD Results, Selected Server Type, SSD Workload, Interface types, Form Factors, Mainstream and Non Mainstream….**

Form Factor Page

**2.5” SFF, 3.5” LFF, Add-In Card, M.2 and M.2 E Kit**

**Next**

Best Availability Page

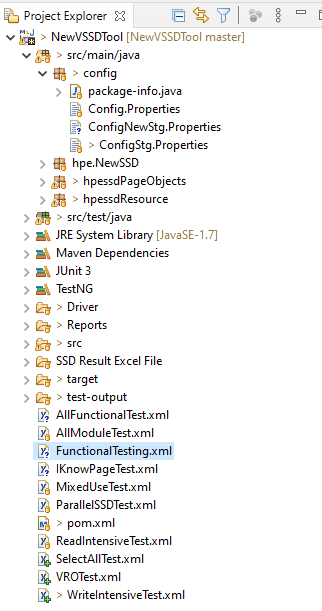
**Mainstream and Non Mainstream**

**Next**

**Automation Framework Design Details**

1. In SSD Selector Tool Project,

* **src/main/java**
  + 1. Package **Config** (it is properties file, which contains all details about the SSD Results and Browser to be used)
       1. In the Configuration file, browser can be changed, to run the test scripts
    2. Package **hpessdPageObjects** (This package consists of all pages of SSD Selector Tool Application)
    3. Package **hpessdResource** (This package consists of BaseClass and SSDTestReport Class)
       1. **BaseClass** class consist of **Initialization** of **Driver** and **Browsers**.
       2. To Capture the **Failed Test screenshot method** written in the **BaseClass**
       3. **SSDTestReport** class consist of Configuration of Result Display
* **src/test/java**
  + 1. Package **hpe.NewSSD** (This package consists of all Test cases based upon the Pages and Individual Test Class files)
* **Driver Folder,** it consists of Browser’s drivers (**Chrome, Firefox** and **Microsoft Edge** Driver) are stored in the Driver Folder**.**
* **Report Folder,** It Consists of Test Results and Failed Test Cases Screen Shots
* **Test-output Folder, It Consist of test results with respect to the**
  1. **Test cases with passed and Failed xml file**
  2. **Emailable-report.xml**
* **POM.xml,** it is the heart of the SSD Selector Tool Project**,** in the POM.xml we add the dependences.



**Automation Framework Implementation**

1. SSD Selector Tool has the following pages in the application,

* **Home Page**
* **Workload Page**
* **SSD type Page**
* **Server type Page**
* **Interface type Page**
* **Form Factor Page**
* **Best Availability Page**
* **Result Page**

1. In each POM Class, I declared the elements present using @**FindBy** annotation.
2. Once all Elements are declared in the POM Class, Initialize the elements declared using, **PageFactory.initElements()** method.
3. Class Files according to the Pages,
4. Based on the No. of Test Cases, we have created the No. of Test Cases in the application.
5. In Test Method, we created object of respective POM Class and using this reference variable, we keep calling the relevant method of POM Class based in the Manual test steps.

**Test Cases and XML File and Execution Details**

1. Implemented the Test cases based on the all pages, which includes all scenarios from the Home Page to Result Page.

* Execution of test cases start from Home Page with “**Help Me Choose”** Option and execution ends in the Result Page.
* **FunctionalTesting**.xml file, In the FunctionalTesting xml file, it covers all functional testing with respect to the individual pages.

1. Implemented the Test case with Selecting the Single Workload, and reaching to the result page.

* **ReadIntensiveTest**.xml
* Covered all test cases with respect to the functionality testing.
* Selecting the Sever type and Server Model.
* In Result Page Validating the test cases, selected configurations are able to see and executed the test cases with pass Conditions.
* In Result Page, Validating the test cases with respect to all functionality,
  + 1. All Labels are displayed correctly
    2. All Slider Values with respect to Min and Max
    3. Valid SKU Search and Invalid SKU Search
    4. Selecting the Server Type and Server Model
    5. Enabling and Disabling the checkbox, SSDs results are displayed
    6. Filtering the check box, according to SSDs results are displayed.
    7. File Download, (Excel, CSV) and Individual File Download
    8. Links.
    9. Validating the Page redirecting
    10. Alert Message Display.

1. Similarly implemented the Test cases by Selecting Single workload for Mixed Use, VRO and Write Intensive. Workloads.

* **MixedUseTest**.xml
* **VROTest.xml**
* **WriteIntensiveTest**.xml

1. Select All Option is selected in all pages, implemented the test case

* SelectAllTest.xml

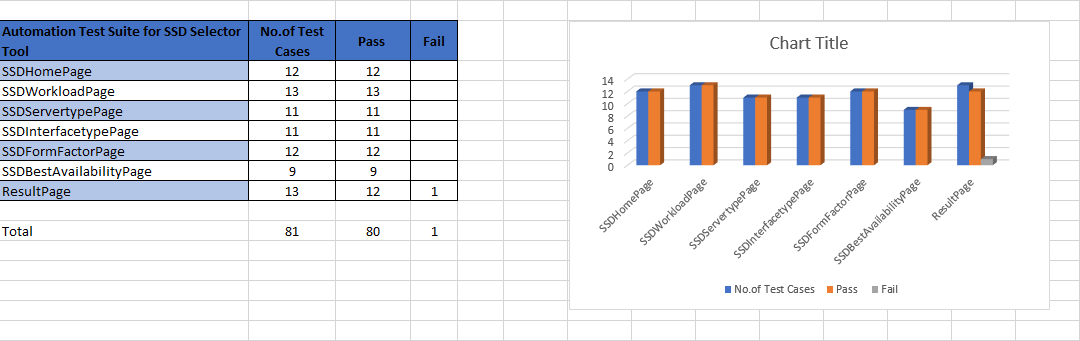
1. Implemented the Test cases for the **I Know What I Need** page,

* In Home Page, selecting with “**I Know What I Need**” Option, and executing all test cases.
* **IKnowPageTest**.xml

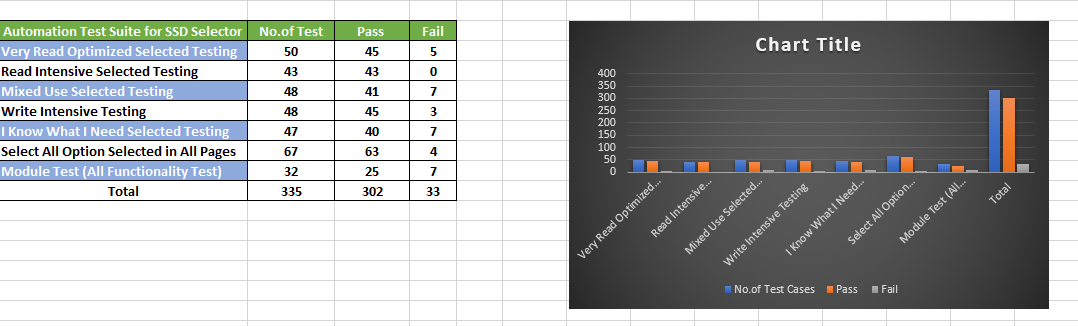
1. Running the test cases by including all single workloads in the one XML File.

* **AllModuleTest**.xml
* **In AllModuleTest xml it has (ReadIntensiveTest**, **MixedUseTest**, **VROTest**, **WriteIntensiveTest**)

**Regression and Integration Testing Report**



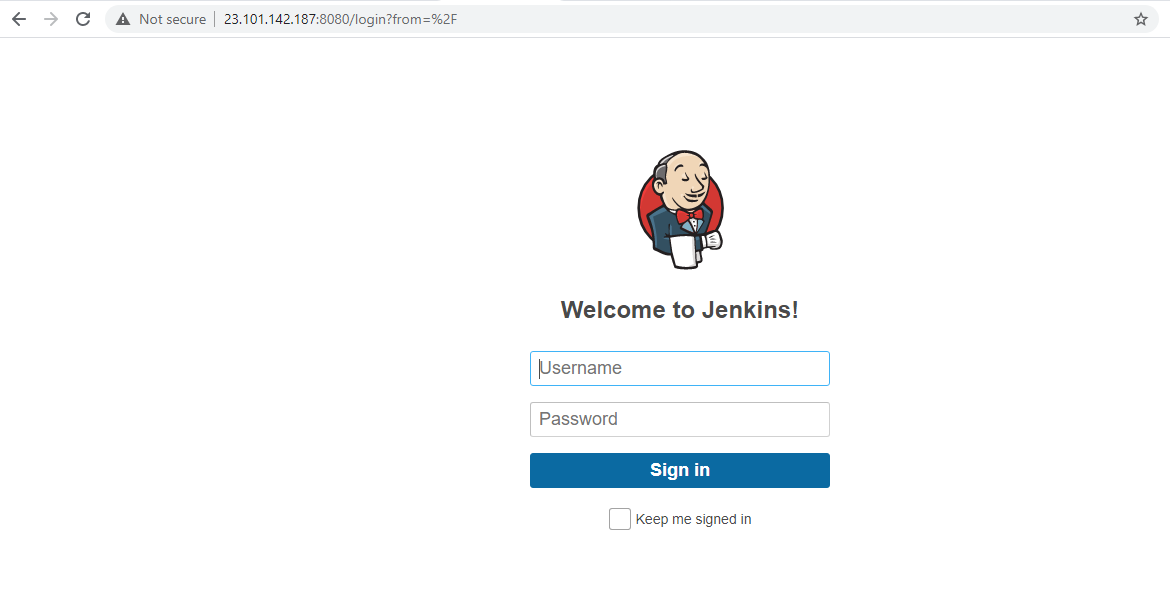
**Module Wise Testing Report**



* **Execution of Test Cases through Jenkins Console**

SSD Selector Automation Testing Project, deployed in the Azure Virtual Machine for running the Test Cases through Jenkins Console.

* Azure Virtual Windows Machine, It Can be connected through rdp. For Running the Jenkins service.
  1. Public IP: - 23.101.142.187
  2. Username: - prashant\_kinagi
  3. Password: - Jenkins@2021WVM
* For Running the Jenkins Service.
  1. Jenkins War deployed in this path “C:/user/prashant\_kinagi/SSDJenkin/”
  2. Open the Command Prompt, and reach to above path
  3. Java –jar Jenkins.war
  4. Jenkins will be started.
* In the Browser,
  1. <http://23.101.142.187:8080/login?from=%2F>
  2. Open the above link for Jenkins Login

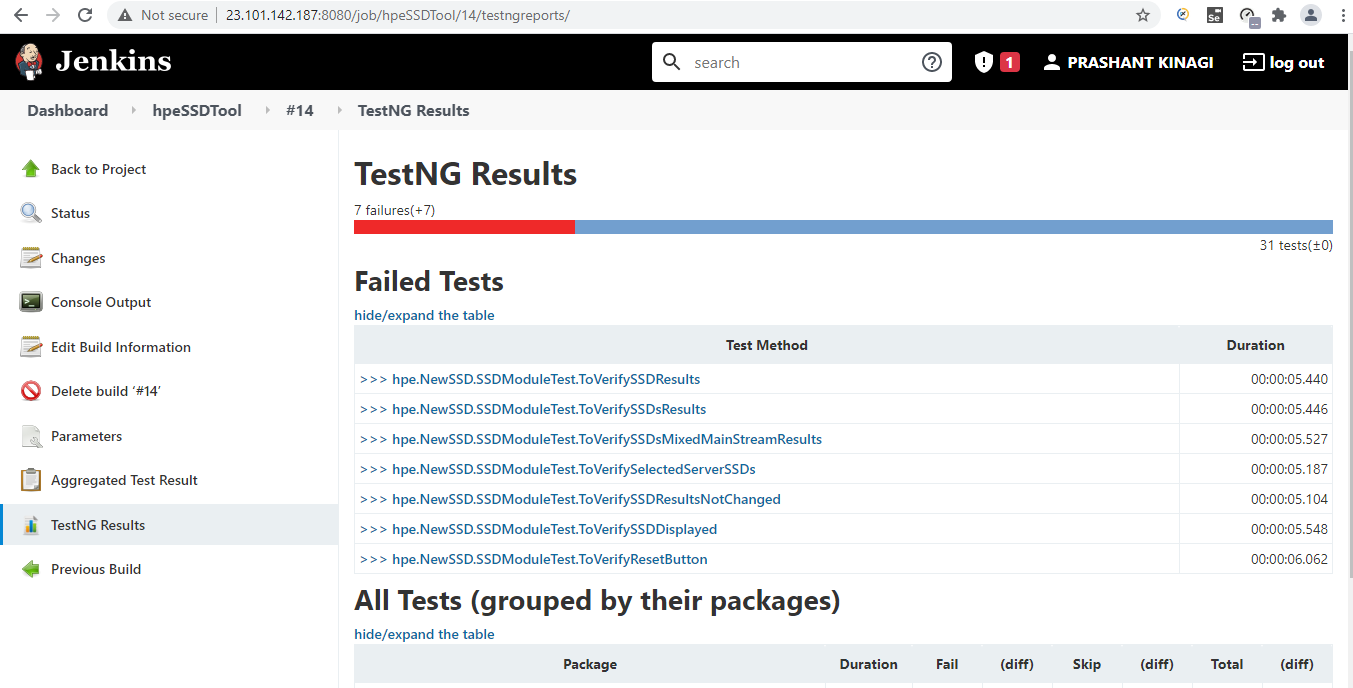


* 1. Username: - hpessdQAAdmin
  2. Password: - hpeSSD@2021
* In the Jenkins Dashboard,
  1. Job Created as “hpeSSDTool”
  2. Click on “hpeSSDTool”,
  3. Click on “Build” option,
  4. In the “Goals” Tab, Mention test want to run through the Jenkins. In the below format

Test –P”IDName” –Dbrowser=”$browser”

**For example**: - **test –PReadWorkloadSelectTest –Dbrowser=”$browser”**

* 1. Click on Save Button.
  2. In the Project “hpeSSDTool” Console,
  3. Click on “**Build with Parameters**” options
  4. Select the browser and click on “**Build** “
  5. Browser can be changed by clicking on drop down. Supported browser are (chrome, MicrosoftEdge and Firefox)
  6. Once the Build Completed. Results can be viewed with respect to Number of test cases and failed.



* Test Id are created
  1. IDName: - **SimpleRegressionTesting**

test –PSimpleRegressionTesting **–**Dbrowser=”$browser”

* + - In this testing, Test cases are executed with respect to the functionality of the SSD Selector Tool, By Selecting the “Mixed Use” Workload and Perform the different operations, reaches to result page and validates the test cases.
  1. IDName: - **SmokeTesting**

test –PSmokeTesting **–**Dbrowser=”$browser”

* + - In this testing, Test cases are executed, From Home Page to Result Page, all test cases are executed in the sequential Manner.
  1. IDName: - **IKnowPageTest**

test -PIKnowPageTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed in the “I Know What I Need” Option, Test cases are validated with respect to the Filtering the Options (SSD Workload, Interface type, Form Factor and Best Availability) and Checking the SSD Results according to the Selections.
  1. IDName: - **ReadWorkloadSelectTest**

test -PReadWorkloadSelectTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “Read Intensive” Workload from the Workload page, performing the different operations, reaches to result page and validates the test cases.
  1. IDName: - **MixedWorkloadSelectTest**

test -PMixedWorkloadSelectTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “MixedUse” Workload from the Workload page, performing the different operations, reaches to result page and validates the test cases.
  1. IDName: - **WriteWorkloadSelectTest**

test -PWriteWorkloadSelectTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “Write Intensive” Workload from the Workload page, performing the different operations, reaches to result page and validates the test cases.
  1. IDName: - **VROWorkloadSelectTest**

test -PVROWorkloadSelectTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “VRO” Workload from the Workload page, performing the different operations, reaches to result page and validates the test cases.
  1. IDName: - **SelectAllFunctionTest**

test -PSelectAllFunctionTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “Select All Option” All Workloads are selected from the Workload page, performing the different operations, reaches to result page and validates the test cases.
  1. IDName: - **AllWorkloadTest**

test -PAllWorkloadTest -Dbrowser="$browser"

* + - In this testing, Test cases are executed by selecting the “Read Intensive” Workload from the Workload page, performing the different operations, reaches to result page and validates the test cases.
    - Similarly Select the **MixedUse, WriteIntensive** and **VRO** Workload test cases are executed by, Selecting One by One workload and performs the testing activities.
* Code has pushed to github
  1. For Cloning the Project, using below link

[git@github.hpe.com:cruz/ssdselectortoolQA.git](mailto:git@github.hpe.com:cruz/ssdselectortoolQA.git)