Robot Framework Level 1

Overview

Installation

Python 3

Pip

Pip install robot-framework

Tags are used to force Test runner with selective run capabilities

Cross-platform abilities(win,mac,linux)

Keyworkd driven (w/ data driven capabilities)

How to bypass select certificate in chrome

How to run test in different browsers using Run Keyword If

Generating allure-report using –listener allure\_robotframework, usage: <https://pypi.org/project/allure-robotframework/>

Using –loglevel trace to log extra details in report

Organization/Directory structure for Robot Framework test

-Tests

-Resources

-common (test setup, tear down etc.)

-keyword files

-po (page objects)

-Results

Sections of Script File: (all these sections are kept in starting \*\*\* and ending \*\*\*)

1. Settings
   1. Documentation - some info about the suite
   2. Library - library being imported
   3. Resource - resources files imported (either keywords, common or po)
   4. Suite Setup/Suite Tear down – some keywords
   5. Test Setup/Test Teardown – some keywords
   6. Test Timeout – X
2. Variables
3. Test Cases
   1. [Documentation] – some info about the test
   2. [Tags] tag 1 tag2
   3. [Timeout] X
4. Keywords (optional – usually called from separate files)

Command line option to run robot files:

robot --loglevel trace -d results --listener allure\_robotframework;results/allure tests/amazon.robot

allure serve results/allure (to generate allue report in temp and open in browser)

* Removing output files on each run
  + Using OperatingSystem library keywords (use it under Suite Setup under setting section)
    - Remove File path to file
    - Remove Files path to first file path to second file …
    - Ex: Remove Files results/allure/\*.\* (will remove all files under allure directory)

**\*\*\* Settings \*\*\*  
Documentation** *This is a sample suite to launch Amazon website****Library*** *SeleniumLibrary****Library*** *OperatingSystem***Suite Setup** Remove Files *results/allure/\*.\****\*\*\* Variables \*\*\*  
  
  
\*\*\* Test Cases \*\*\*  
Launch Amazon Website  
 [Documentation]** *This is sample test case to Launch Amazon website* **[Tags]** *This is a smoke test* Open Browser *http://amazon.in ff* **${title}**= get title  
 Should Contain ${title} *Online Shopping Site ignore\_case=True* Close Browser  
  
  
**\*\*\* Keywords \*\*\***

------------------------------------------------------------------------------------------------------------------------------------------

**\*\*\* Settings \*\*\*  
Documentation** *saucedemo e-commerce application automation****Library*** *SeleniumLibrary****Library*** *OperatingSystem***Suite Setup** Remove Files *results/allure/\*.\* results/\*.png***Suite Teardown** Close Browser  
  
  
**\*\*\* Variables \*\*\*  
${username}**= *standard\_user***${password}**= *secret\_sauce***${firstname}**= *Alex***${lastname}**= *Casto***${postalcode}**= *160062***\*\*\* Test Cases \*\*\*  
Launch SauceDemo Website  
 [Documentation]** *SauceDemo Homepage should be loaded* **[Tags]** *Smoke* Open Browser *https://www.saucedemo.com ff* **${title}**= get title  
 Should Contain ${title} *Swag ignore\_case=True***Login with Correct Credentials  
 [Documentation]** *User should be able to login successfully* **[Tags]** *Login* Input Text *id=user-name* ${username}  
 Input Password *id=password* ${password}  
 Press Keys *id=password RETURN* Wait Until Page Contains *Products* **${curr\_url}**= Get Location  
 Should Contain ${curr\_url} *inventory***Select an Item and add to cart  
 [Documentation]** *Add item to cart from product page* **[Tags]** *AddToCart* Click Button *ADD TO CART* Click Element *xpath=//\*[@id="shopping\_cart\_container"]/a* Wait Until Page Contains *REMOVE* Page Should Contain *CHECKOUT***Proceed to Checkout  
 [Documentation]** *Proceeding for checkoout* **[Tags]** *Checkout* Click Element *xpath=//\*[@id="cart\_contents\_container"]/div/div[2]/a[2]* Wait Until Page Contains *Checkout: Your Information* Input Text *id=first-name* ${firstname}  
 Input Text *id=last-name* ${lastname}  
 Input Text *id=postal-code* ${postalcode}  
 Click Button *CONTINUE* Wait Until Page Contains *Checkout: Overview* Click Element *xpath=//\*[@id="checkout\_summary\_container"]/div/div[2]/div[8]/a[2]* Wait Until Page Contains *THANK YOU FOR YOUR ORDER* Page Should Contain *Your order has been dispatched, and will arrive***Logout the user  
 [Documentation]** *User should be logout* **[Tags]** *Logout* Click Button *Open Menu* Wait Until Element is Visible *logout\_sidebar\_link* Click Link *logout\_sidebar\_link* **${main\_url}**= Get location  
 *#Wait Until Page Contains Accepted usernames are:* Should Be Equal As Strings ${main\_url} *https://www.saucedemo.com/index.html***\*\*\* Keywords \*\*\***

* Running specific tags via commandline
  + Robot -d results –include smoke tests/amazon.robot
  + robot -d results -i smoke tests/amazon.robot
    - if you want to include more than one tag to run you have to use like

-i tag1 -i tag2

* + robot -d results -v Browser:ie tests/amazon.robot
* Renaming Default Suite/Parent level Suite name to include in report to other
  + Robot -d results -N “Full Regression” tests (all folders and including robot files will be executed) or you can say running multiple suite files
  + ex: robot --loglevel trace -d results --listener allure\_robotframework;results/allure -N "Full Regression" tests
* Running a specific test case
  + robot --loglevel trace -d results --listener allure\_robotframework;results/allure -N "Full Regression" -t "Launch SauceDemo Website" tests
  + if you want to run more than one test cases using ‘-t’ you have to pass ‘-t’ for each test case you want to run for ex: robot -d results -t “test case1” -t “test case 2” tests

------------------------------------------------------------------------------------------------------------------------------------------

Using keywords from this file to use under Suite Setup and Suite Teardown

**\*\*\* Settings \*\*\*  
*Library*** *OperatingSystem***\*\*\* Keywords \*\*\*  
Clean Stuff Before Test Run** *#Remove Files results/allure/\*.\* results/robot-report/\*.png* Remove Files *Results/allure-results/\*.\** Remove Directory *Results/allure-report recursive=yes  
 #Remove Files Results/robot-report/\*.png* Remove Files *Results/robot-report/\*.png Results/robot-report/\*.html Results/robot-report/\*.log***Post Run Steps** Close Browser  
 Run *allure generate results/allure-results -o results/allure-report* Copy Files *results/robot-report/\*.png results/allure-report/data/attachments*

------------------------------------------------------------------------------------------------------------------------------------------

Calling in another file

**\*\*\* Settings \*\*\*  
Documentation** *This is a sample suite to launch Amazon website****Library*** *SeleniumLibrary****Library*** *OperatingSystem****Resource*** *../Resources/Cleaning.robot***Suite Setup** Clean Stuff Before Test Run

**Suite Teardown** Post Run Steps

Test Setup

Test Teardown

------------------------------------------------------------------------------------------------------------------------------------------

Logging thigs in robot framework:

Log to console log this to console output

Log log this to report (this will logged directly in report of test results not on console)

Variables in Robot Framework:

${myvar} = Set Variable testvalue (Set Variable is used when we have to use variable explicitly in Test Case only)

Log ${myvar} (will print – testvalue)

* Scalare variable (as declared above hold single value)
* List Variable (holds multiple values)
  + @{mylistvar} = test1 test2 test 3 test4
  + @{mylistvar} = Set Variable test1 test2 test 3 test4 (instead of using Set Variable for List Variable use “Create List” keyword instead)
  + Log @{mylistvar}[1]

Naming Convention used for Global/nonglobal variables: Global variables are represented by CAPS and nonglobal as SMALL-CASE. Global variables are declared in \*\*\* Variables \*\*\* section

Global variable ex: ${BROWSER} ie

${BASE\_URL} <http://www.hello.com>

Precedence of variable:

Command line > Script > Keyword file

Overriding Existing Variables in script using commandlin:

* + robot -d results -v BROWSER:ie -v SEARC\_TRM:”Iphone” tests/amazon.robot

Passing variables to keywords for making reusable keywords:

Begin Web Test <http://www.hello.com> ie

\*\*\* Keywords \*\*\*

Begin Web Test

[Arguments] ${url} ${browser}

Open browser ${url} ${browser}

Using List Variable:

@{url\_and\_browser} = Create List or Set Variable <http://www.hello.com> ie

Begin Web Test @{url\_and\_browser}

\*\*\* Keywords \*\*\*

Begin Web Test

[Arguments] @{url\_and\_browser}

Open browser @{url\_and\_browser}[0] @{url\_and\_browser}[1]

Where to get support for Robot framework questions: <https://robotframework.org/#community> . You can go through slack or other option to connect

Find usage : this is about to check where a particular code chunk is linked to other files, select the line in code > right click > find usages

Use fo TODO Panel: if we have to mark in script files that we want to work on later or to remind our self that something is pending, we can define TODO in script file and when we come again in PyCharm and will click on TODO panel it’ll show us what’s pending

Ex: # TODO move this element into a variable

Changing Report and Log Titles: robot --loglevel trace -d results/robot-report --listener allure\_robotframework;results/allure-results --reporttitle "Demoblaze Test

Report" --logtitle "Applicaton Logs" -N "Full Test" tests/Sign\*

By default all tests are considered as critical, but we can specify which test case to be considered as critical by specifying tag with –critical or -c option:

Robot -d results -c tag1 -c tag2 tests/hello.robot

Using non-critical does work vice a versa using –noncritical or -n option:

Robot -n tag1 -n tag2 tests/hello.robot

Preventing test result/log to be overridden using –timestampoutputs or -T option and it will generate separate results/log output files on each run with timestamp

Robot -n tag1 -n tag2 -T tests/hello.robot

Forcing script execution order : by default normally executes in alpha order but if you want to execute them in order you can prefix file name with xx\_

Ex: 01\_some\_feature.robot – this will execute first

02\_other\_feature.robot – this will execute after first one

And generated report will not show prefixes (01, 02…)

Executing scripts in random order using –randomize flag: can use aforesaid command flag

Tests

Suites

All

None

Setting Log Level form Script instead command line: use set log level <trace/debug/info..>

Automatic Variables: to use them in informative manner or for conditional purpose, go to robot framework user guide and search for automatic variable: <https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#automatic-variables>

How to return a value through a keyword:

Using [RETURN] we can specify what to return by keyword

Ex: user keyword

Log this

${var} = get location

Log this

[RETURN] ${var}

In another script we can store in a variabl like:

${somevar} = user keyword

Log ${somevar} (it will print – logged)

Learn python by hard way to learn python: <https://learnpythonthehardway.org/> scroll to the end to check for free content

http://www.Pypi.python.org list out all the packages

Locators: xpath and css strategies

For skipping middle tags in xpath we just user // but in css we use a space instead

Ex: xpath: //div[@id=’temp’]/div//input//a

Css: div[id=’temp] > div input a

Xpath: //h2/span/img[@alt=’test’]

Css: h2 > span > img[alt=’test’]

Starts with:

Xpath: //input[starts-with(@type,’submit’)] for type=submit-238839 something like that

Css: input[type^=’submit’]

Ends with:

Xpath: //input[ends-with(@type,’submit’)] for type=submit-238839 something like that

Css: input[type$=’submit’]

Contains:

Xpath: //input/a[contains(text(),’Full or partial link text’)] used for elements have text inside the tag, for hyperlinks, divs, textareas etc

OR

Xpath: //input/a[contains(.,’Full or partial link text’)]

Xpath: //input[contains(@type,’submit’)] for type=submit-238839 something like that

Css: input[type\*=’submit’]

**How to break a long line of code into multiple lines using eliipses(…)**

Kyeworkd 1 k dk dfdf fdf sd dfdf dfdsfdsf sfdf dfdsf fdsfdf dfdfdf sf

… keruud jkkjjkj

… kdfkd fdsjf dfdfd

Using IF ELSE:

using builtin keywords

Run Keyword IF <condition> keyword1

Run Keyword IF <condition> keyword1 ELSE keyword2

Run Keyword IF <condition> keyword1 ELSE IF <condition> keyword2 ELSE keyword3

**\*\*\* Settings \*\*\*  
*Library*** *BuiltIn***\*\*\* Variables \*\*\*  
${marvar}** = *2***\*\*\* Test Cases \*\*\*  
IFElseDemo** Run Keyword If ${marvar} *> 3 keyword1* Run Keyword If ${marvar} *> 3 keyword1 ELSE keyword2* Run Keyword If ${marvar} *> 3 keyword1 ELSE IF* ${marvar} *< 3 keyword2 ELSE keyword3***\*\*\* Keywords \*\*\*  
keyword1** log *iamin1***keyword2** log *iamin2***keyword3** log *iamin3*

Using FOR:

For Loop in Range:

**\*\*\* Test Cases \*\*\*  
Only upper limit  
 [Documentation]** *Loops over values from 0 to 9* FOR ${index} *IN RANGE 10* Log ${index}  
 END  
  
**Start and end  
 [Documentation]** *Loops over values from 1 to 10* FOR ${index} *IN RANGE 1 11* Log ${index}  
 END  
  
**Also step given  
 [Documentation]** *Loops over values 5, 15, and 25* FOR ${index} *IN RANGE 5 26 10* Log ${index}  
 END  
  
**Negative step  
 [Documentation]** *Loops over values 13, 3, and -7* FOR ${index} *IN RANGE 13 -13 -10* Log ${index}  
 END  
  
**Arithmetic  
 [Documentation]** *Arithmetic with variable* FOR ${index} *IN RANGE* ${var} *+ 1* Log ${index}  
 END  
  
**Float parameters  
 [Documentation]** *Loops over values 3.14, 4.34, and 5.54* FOR ${index} *IN RANGE 3.14 6.09 1.2* Log ${index}  
 END

For looping over a list:

**Looping through a general list** FOR ${item} *IN item1 item2 item3 item4* Log *I am logging* ${item} *from general list* END  
  
**Looping through a list variable  
 @{mylist}** create list *item1 item2 item3 item4 item5* FOR ${item} *IN* @{mylist}  
 Log *I am logging* ${item} *from list variable* END  
  
**Exiting and Continue from FOR loop  
 @{mylist}** create list *item1 item2 item3 item4 item5* FOR ${item} *IN* @{mylist}  
 Log *I am logging* ${item} *from list variable* Run Keyword If *"*${item}*" == "item3" exit for loop* Log *Not yet exited* continue for loop  
 END  
  
**For-in-enumerate  
 @{mylist}** create list *item1 item2 item3 item4 item5* FOR ${index} ${item} *IN ENUMERATE* @{mylist}  
 Log *item at index*${index} *is* ${item}  
 END

**Uses of Dictionaries: dictionary works same as map in java it works on key value basis**

From commandline: robot -d results/robot-report -v BROWSER:ff -v ENV:qa tests/saucedemoG\*

Script:

**\*\*\* Settings \*\*\*  
Documentation** *Saucedemo e-commerce application automation****Resource*** *../Resources/SaucedemoUI.robot****Resource*** *../Resources/Common.robot***Test Setup** Begin Web Test ${ENV\_URLS} ${BROWSER}  
**Test Teardown** End Web Test  
**Suite Teardown** Generate Allure Report  
  
**\*\*\* Variables \*\*\****#use of dictionary variable  
&{cust\_shipping\_details} = firstname=Alex lastname=Casto postalcode=160062  
&{ENV\_URLS} = dev=https://www.saucedemo.com qa=http://www.saucedemo.com prod=https://www.saucedemo.com***${BROWSER}** *chrome***${ENV}** *qa  
  
#user login data  
&{valid\_user} = username=standard\_user password=secret\_sauce  
&{invalid\_user} = username=hellouser password=hellopassword***\*\*\* Test Cases \*\*\*  
Verify An Invalid User Is Not Able To Login  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* Login As An Invalid User ${invalid\_user}  
  
  
**Verify E2E Product Order Flow For A Valid User  
 [Documentation]** *E2E Product Order Flow For A Valid User* **[Tags]** *Smoke PostivieScenario* Given Verify Title Of The Landed Page  
 When Login As A Valid User ${valid\_user}  
 Then Verify Logged In Successfully  
 When Select An Item And ADD It To The Cart  
 Then Verify Cart Page Gets Loaded  
 When Proceed To Checkout & Verify Checkout Page Loaded  
 And Enter Shipping Details & Continue ${cust\_shipping\_details}  
 Then Verify Checkout:Overview Page Loaded & Continue To Place Order  
 And Verify Order Confirmation Page Loaded  
 When Logout From The SauceDemo Website  
 Then Verify Logout Successfully

Common.robot

**\*\*\* Settings \*\*\*  
*Library*** *SeleniumLibrary****Library*** *OperatingSystem***\*\*\* Keywords \*\*\*  
Begin Web Test  
 [Arguments] ${env\_url} ${browser}** Open Browser ${env\_url.${ENV}} ${browser}  
 Maximize Browser Window  
  
**End Web Test** Close Browser  
  
**Generate Allure Report** Run *allure generate results/allure-results -o results/allure-report* Copy Files *results/robot-report/\*.png results/allure-report/data/attachments*

Checkout Preview.robot

**\*\*\* Settings \*\*\*  
*Library*** *SeleniumLibrary***\*\*\* Variables \*\*\*  
${CHK\_PRV\_FIRST\_NAME}** *id=first-name***${CHK\_PRV\_LAST\_NAME}** *id=last-name***${CHK\_PRV\_PSTL\_CD}** *id=postal-code***\*\*\* Keywords \*\*\*  
Checkout Preview Page Loaded** Wait Until Page Contains *Checkout: Your Information***Enter First Name  
 [Arguments] ${cust\_ship\_data}** Input Text ${CHK\_PRV\_FIRST\_NAME} ${cust\_ship\_data.firstname}  
  
**Enter Last Name  
 [Arguments] ${cust\_ship\_data}** Input Text ${CHK\_PRV\_LAST\_NAME} ${cust\_ship\_data.lastname}  
  
**Enter Postal Code  
 [Arguments] ${cust\_ship\_data}** Input Text ${CHK\_PRV\_PSTL\_CD} ${cust\_ship\_data.postalcode}  
  
**Continue To Order Preview** Click Button *CONTINUE*

UI.robot

**\*\*\* Settings \*\*\*  
*Resource*** *./PO/Login.robot****Resource*** *./PO/Homepage.robot****Resource*** *./PO/Cart.robot****Resource*** *./PO/CheckoutPreview.robot****Resource*** *./PO/OrderReview.robot****Resource*** *./PO/OrderConfirmation.robot***\*\*\* Variables \*\*\****#Valid Login Credentials***${username}**= *standard\_user***${password}**= *secret\_sauce  
  
#Invalid Login Credentials***${inv\_user}** *hello***${inv\_passwd}** *there  
  
#shipping details info***${firstname}**= *Alex***${lastname}**= *Casto***${postalcode}**= *160062***\*\*\* Keywords \*\*\*  
Verify Title Of The Landed Page** Loginpage Loaded  
  
**Login As A Valid User  
 [Arguments] ${valid\_user\_credentials}** Enter Username ${valid\_user\_credentials}  
 Enter Password ${valid\_user\_credentials}  
 Submit Login Form With Credentials  
 Wait For Homepage Load After Login  
  
**Login As An Invalid User  
 [Arguments] ${invalid\_user\_credentials}** Enter Username ${invalid\_user\_credentials}  
 Enter Password ${invalid\_user\_credentials}  
 Submit Login Form With Credentials  
 Wait For Error Message On Login Page  
  
**Verify Logged In Successfully** Homepage Loaded  
  
**Select An Item And ADD It To The Cart** Click On Add To Cart  
 GoTo Cart Page  
  
**Verify Cart Page Gets Loaded** Cart Page Loaded  
  
**Proceed To Checkout & Verify Checkout Page Loaded** Click On Checkout Button  
 Checkout Preview Page Loaded  
  
**Enter Shipping Details & Continue  
 [Arguments] ${cust\_ship\_data}** Enter First Name ${cust\_ship\_data}  
 Enter Last Name ${cust\_ship\_data}  
 Enter Postal Code ${cust\_ship\_data}  
 Continue To Order Preview  
  
**Verify Checkout:Overview Page Loaded & Continue To Place Order** Order Review Page Loaded  
 Continue To Confirm Order  
  
**Verify Order Confirmation Page Loaded** Order Confirmation Page Loaded  
  
**Logout From The SauceDemo Website** Finding Logout Link & Clicking On It  
  
**Verify Logout Successfully** Loginpage URL Loaded

Login.robot

**\*\*\* Settings \*\*\*  
*Library*** *SeleniumLibrary***\*\*\* Variables \*\*\*  
${LOGIN\_USERNAME\_FIELD}** *id=user-name***${LOGIN\_PASSWORD\_FIELD}** *id=password***\*\*\* Keywords \*\*\*  
Enter Username  
 [Arguments] ${user\_login\_data}** Input Text ${LOGIN\_USERNAME\_FIELD} ${user\_login\_data.username}  
  
**Enter Password  
 [Arguments] ${user\_login\_data}** Input Password ${LOGIN\_PASSWORD\_FIELD} ${user\_login\_data.password}  
  
**Submit Login Form With Credentials** Press Keys ${LOGIN\_PASSWORD\_FIELD} *RETURN***Wait For Error Message On Login Page** Wait Until Page Contains *Username and password do not match any user in this service***Loginpage Loaded  
 ${title}**= get title  
 Should Contain ${title} *Swag ignore\_case=True***Loginpage URL Loaded  
 ${main\_url}**= Get location  
 *#Wait Until Page Contains Accepted usernames are:* Should Be Equal As Strings ${main\_url} *https://www.saucedemo.com/index.html*

**Working With Test Templates:** Used to run a test case with multiple set of test data for example validating login functionality with valid and invalid credentials combination. You can say it a data driven style

First example:

**\*\*\* Settings \*\*\*  
Documentation** *Saucedemo e-commerce application automation****Resource*** *../Resources/SaucedemoUI.robot****Resource*** *../Resources/Common.robot***Test Setup** Begin Web Test ${ENV\_URLS} ${BROWSER}  
**Test Teardown** End Web Test  
**Suite Teardown** Generate Allure Report  
  
**\*\*\* Variables \*\*\****#use of dictionary variable  
&{cust\_shipping\_details} = firstname=Alex lastname=Casto postalcode=160062  
&{ENV\_URLS} = dev=https://www.saucedemo.com qa=http://www.saucedemo.com prod=https://www.saucedemo.com***${BROWSER}** *chrome***${ENV}** *qa  
  
#user login data  
&{valid\_user} = username=standard\_user password=secret\_sauce  
&{invalid\_password} = username=standard\_user password=hellopassword  
&{invalid\_user} = username=hellouser password=secret\_sauce  
&{blank\_user} = username= password=secret\_sauce  
&{blank\_password} = username=standard\_user password=  
&{blank\_user\_passwd} = username= password=***\*\*\* Test Cases \*\*\*  
Verify Login Should Fail For Invalid Credentials  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* **[Template]** Invalid Login Credentials Should Fail Login  
 **${invalid\_password}  
  
  
Verify Login Should Fail For Invalid Credentials  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* **[Template]** Invalid Login Credentials Should Fail Login  
 **${invalid\_user}  
  
Verify Login Should Fail For Invalid Credentials  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* **[Template]** Invalid Login Credentials Should Fail Login  
 **${blank\_user}  
  
Verify Login Should Fail For Invalid Credentials  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* **[Template]** Invalid Login Credentials Should Fail Login  
 **${blank\_password}  
  
Verify Login Should Fail For Invalid Credentials  
 [Documentation]** *Verification of Negative Testing Scenario For Login* **[Tags]** *NegativeScenario* **[Template]** Invalid Login Credentials Should Fail Login  
 **${blank\_user\_passwd}  
  
\*\*\* Keywords \*\*\*  
Invalid Login Credentials Should Fail Login  
 [Arguments] ${invalidcredentials}** Login As An Invalid User ${invalidcredentials}

@2nd Example: More better

**\*\*\* Settings \*\*\*  
Documentation** *Saucedemo e-commerce application automation****Resource*** *../Resources/SaucedemoUI.robot****Resource*** *../Resources/Common.robot***Test Setup** Begin Web Test ${ENV\_URLS} ${BROWSER}  
**Test Teardown** End Web Test  
**Suite Teardown** Generate Allure Report  
**Test Template** Invalid Login Credentials Should Fail Login  
  
**\*\*\* Variables \*\*\****#use of dictionary variable  
&{cust\_shipping\_details} = firstname=Alex lastname=Casto postalcode=160062  
&{ENV\_URLS} = dev=https://www.saucedemo.com qa=http://www.saucedemo.com prod=https://www.saucedemo.com***${BROWSER}** *chrome***${ENV}** *qa  
  
#user login data, updated dictionaries to include error message in every case to validate on invalid login  
&{valid\_user} = username=standard\_user password=secret\_sauce  
&{invalid\_password} = username=standard\_user password=hellopassword error\_msg=Username and password do not match any user in this service  
&{invalid\_user} = username=hellouser password=secret\_sauce error\_msg=Username and password do not match any user in this service  
&{invalid\_user\_passwd} = username=hellouser password=hellopasswd error\_msg=Username and password do not match any user in this service  
&{blank\_user} = username= password=secret\_sauce error\_msg=Username is required  
&{blank\_password} = username=standard\_user password= error\_msg=Password is required  
&{blank\_user\_passwd} = username= password= error\_msg=Username is required*

\*\*\* Test Cases \*\*\* INVALID\_USER\_CREDENTIALS  
*Login With Invalid User* ${invalid\_user}  
*Login With Invalid Password* ${invalid\_password}  
*Login With Blank User* ${blank\_user}  
*Login With Blank Password* ${blank\_password}  
*Login With Blank User And Password* ${blank\_user\_passwd}  
  
  
**\*\*\* Keywords \*\*\*  
Invalid Login Credentials Should Fail Login  
 [Arguments] ${invalidcredentials}** Login As An Invalid User ${invalidcredentials}

Login.robot used in SauceDemoUI.robot file

**\*\*\* Settings \*\*\*  
*Library*** *SeleniumLibrary***\*\*\* Variables \*\*\*  
${LOGIN\_USERNAME\_FIELD}** *id=user-name***${LOGIN\_PASSWORD\_FIELD}** *id=password***\*\*\* Keywords \*\*\*  
Enter Username  
 [Arguments] ${user\_login\_data}** Input Text ${LOGIN\_USERNAME\_FIELD} ${user\_login\_data.username}  
  
**Enter Password  
 [Arguments] ${user\_login\_data}** Input Password ${LOGIN\_PASSWORD\_FIELD} ${user\_login\_data.password}  
  
**Submit Login Form With Credentials** Press Keys ${LOGIN\_PASSWORD\_FIELD} *RETURN***Wait For Error Message On Login Page  
 [Arguments] ${user\_login\_data}** Wait Until Page Contains ${user\_login\_data.error\_msg}  
  
**Loginpage Loaded  
 ${title}**= get title  
 Should Contain ${title} *Swag ignore\_case=True***Loginpage URL Loaded  
 ${main\_url}**= Get location  
 *#Wait Until Page Contains Accepted usernames are:* Should Be Equal As Strings ${main\_url} *https://www.saucedemo.com/index.html*

**Calling Variables from other file:**

InputData.robot:

**\*\*\* Variables \*\*\****#use of dictionary variable  
&{cust\_shipping\_details} = firstname=Alex lastname=Casto postalcode=160062  
&{ENV\_URLS} = dev=https://www.saucedemo.com qa=http://www.saucedemo.com prod=https://www.saucedemo.com***${BROWSER}** *chrome***${ENV}** *qa  
  
#user login data  
&{VALID\_USER} = username=standard\_user password=secret\_sauce  
&{INVALID\_PASSWORD} = username=standard\_user password=hellopassword error\_msg=Username and password do not match any user in this service  
&{INVALID\_USER} = username=hellouser password=secret\_sauce error\_msg=Username and password do not match any user in this service  
&{INVALID\_USER\_PASSWD} = username=hellouser password=hellopasswd error\_msg=Username and password do not match any user in this service  
&{BLANK\_USER} = username= password=secret\_sauce error\_msg=Username is required  
&{BLANK\_PASSWORD} = username=standard\_user password= error\_msg=Password is required  
&{BLANK\_USER\_PASSWD} = username= password= error\_msg=Username is required*

Script File where variables were called:

**\*\*\* Settings \*\*\*  
Documentation** *Saucedemo e-commerce application automation****Resource*** *../Resources/SaucedemoUI.robot****Resource*** *../Resources/Common.robot****Resource*** *../Resources/InputData.robot***Test Setup** Begin Web Test ${ENV\_URLS} ${BROWSER}  
**Test Teardown** End Web Test  
**Suite Teardown** Generate Allure Report  
**Test Template** Invalid Login Credentials Should Fail Login  
  
  
\*\*\* Test Cases \*\*\* INVALID\_USER\_CREDENTIALS  
Login With Invalid User And Password ${INVALID\_USER\_PASSWD}  
Login With Invalid User ${INVALID\_USER}  
Login With Invalid Password ${INVALID\_PASSWORD}  
Login With Blank User ${BLANK\_USER}  
Login With Blank Password ${BLANK\_PASSWORD}  
Login With Blank User And Password ${BLANK\_USER\_PASSWD}  
  
  
**\*\*\* Keywords \*\*\*  
Invalid Login Credentials Should Fail Login  
 [Arguments] ${invalidcredentials}** Login As An Invalid User ${invalidcredentials}