**VEC-HE Test Automation**

**Here are the Pre-requisites of the framework:**

**System Variables in windows**

**PYTHONPATH**

C:\Python27;C:\Python27\Lib\site-packages;

<Project base directory>;

<Project base directory>\library;

**PATH** should be extended as

C:\Python27;C:\Python27\Scripts;

**Packages to be installed:**

* Pyyaml
* requests
* robotframework
* selenium
* robotframework-selenium2library
* service

**Selenium Automation:**

**Selenium automation can be performed in two different ways:**

* Automation using robotframework-selenium2library
* Automation using python Selenium Web driver.

In this framework both the methods are implemented and clear description is included below:

**Automation using python Selenium Web driver:**

**Framework Structure:**

veche-ta

robot\_tests -> python\_web\_automation

workflow -> baseworkflow

auc

utils

conf

results

**Base use case**:

* It is a template which defines the AUCs functional order.
* Below is the functional order of AUC:
* validate\_context
  + Validates the inputs provided for the AUC and any specific conditions.
* run\_Test
  + Runs the Actual Code for the AUC
* finalize\_context
  + Updates the context with the Required Results of AUC

**Associate use cases (AUC):**

* All test functionality as AUCs going to keep in this folder. All AUCs will inherit functionality from baseusecase

**Utils**:

This folder contain any required utils for the framework like rest api, yaml parser.

* Context management
  + To read and maintain the structure of yaml data

**Conf:**

* This folder contains input parameters which will be required for test execution. Preferably yaml files.

**robot\_tests:**

* This folder contains the Created Robot Suite files to test the UI.

**Workflows:**

* This will contain the workflow for all the test methods.

**Below is the Automation process:**

1. Config file Creation/Update input parameters in the YAML file
2. Inherit functionality from baseusecase and create AUC for each module.
3. Update workflow file with AUC Created
4. Robot file Creation

**AUC Creation**

* First check any AUC is there with the functionality otherwise create AUC.
* To create any new user story as AUC these are the steps to be followed
  + First create a folder under auc\executables on with auc name

**Config file Creation/Update**

* It is used for running the functional tests.

**Update workflow file with the AUC Created**

* We need to update the workflow with the AUC implemented to work with robot

**Robot File Creation:**

Create a Robot file with your test cases and run it.

* **Import baseworkflow libraries into the robot file**

Eg:

**\*\*\* Settings \*\*\***

***Library*** *workflow.baseworkflow.BaseWorkflow*

* **Define the path of YAML file in the variables**

Eg:

**\*\*\* Variables \*\*\***

**${CONF DIR}** */usr/home/robot/generic.yaml*

* Create a testcase by calling procedures defined in **baseworkflow** file.

Eg

**\*\*\* Test Cases \*\*\*  
 UI LOGIN****[Setup]** Apply Settings From Files *C:\\vhche-ta\\conf\\generic.yaml* User Opens Browser  
 User Closes Browser  
 **[Teardown]** Reset Settings

**Automation using Robotframework-selenium2library:**

robot

resource.robot

TC1.robot

TC2.robot

**resource.robot:**

This is the main input file which includes all libraries and inputs.

**Design of resource.robot file:**

**Import libraries:**

We have to import all selenium libraries from **Selenium2Library**

**Eg:**

**\*\*\* Settings \*\*\***

***Library*** *Selenium2Library*

**Define the variables:**

All variables including browser type and url should declare in the robot file

Eg:

**\*\*\* Variables \*\*\***

**${SERVER}** *localhost:3000***${BROWSER}** *Firefox*

**Keywords:**

Procedures need to implement here using Selenium2Library.

**\*\*\* Keywords \*\*\*  
Open Browser To Login Page** Open Browser ${LOGIN URL} ${BROWSER}  
 Maximize Browser Window

Below is the link to know more about **Selenium2Library** functions

Links:

<http://robotframework.org/Selenium2Library/Selenium2Library.html#Click%20Element>

**TC.robot:**

Using resource.robot file, we have to design a TC to execute.

**Design of TC.robot file:**

**Settings:**

**Import all resources from resource.robot file**

**Eg:**

**\*\*\* Settings \*\*\***

***Resource*** *resource.robot*

**Test Cases:**

Create a testcase by calling functions defined in resource.robot file.

Eg:

**\*\*\* Test Cases \*\*\*  
Valid Login** Given browser is opened to login page  
 When user "demo" logs in with password "mode"  
 Then welcome page should be open

**Robot file execution:**

We can execute robot file from terminal using the command:

robot –L DEBUG –d <output file path> <TC.robot>

* **-L DEBUG :** Using this option errors will capture in the report (log.html) file
* **-d <o/p path>:** This command will generate html reports in the specified path.

Eg:

robot –L DEBUG -d C:\Users\madduv\PycharmProjects\vhche-ta\results validate\_page2.robot

**REST API Automation:**

**Framework Structure:**

veche-ta

robot\_tests -> rest\_api\_automation

workflow -> baseworkflow\_rest.py

auc -> rest

utils -> service

conf

results

Automation procedure:

**Step1**: Create restConstants.py file under **conf** directory and include REST APIs related inputs in it.

**Step2**: Create rest.yaml file under **conf** directory and include REST API server and other information in it.

**Step3**: Create a context utility under **utils** directory to convert yaml data (step 2) into dictionary.

**Step4**: Create rest library under **utils**-> **service** directory and add REST API modules in it (example: get, put, post, get\_vmid\_by\_name) with the help of restConstants.py (step1)

**Step5**: By inheriting functionality using baseusecase, create AUC under **auc**->**rest** directory and create REST testcases.

**Step6**: Create baseworkflow\_rest.py under **workflow** directory and perform the following actions:

* Create separate input variables (\_GC\_TAG, \_WORKFLOW\_TAG) to store the data of all yaml files.
* Using context utility (step3), convert the yaml data into dictionary and store the data in input variable (ctx).
* Call all AUCs in the workflow file, and also pass the input to AUCs from here.
* Create a proc to reset all variables to none at the end of the testcase execution.

**Step7**: Create a robot file under directory (robot\_tests -> rest\_api\_automation) and import baseworkflow\_rest python file (step 6) and run the modules included in the workflow.