**REST API Test Automation framework**

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

**System Variables in windows**

**PYTHONPATH**

C:\Python36;C:\Python36\Lib\site-packages;

<Project base directory>;

<Project base directory>\library;

**PATH** should be extended as

C:\Python36;C:\Python36\Scripts;

**Packages to be installed:**

* Pyyaml
* requests
* robotframework
* service

**Framework Structure:**

rest-api-project

robot\_tests -> rest\_api\_automation

workflow -> baseworkflow\_rest.py

auc -> rest

utils -> service

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*

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

Eg

**\*\*\* Test Cases \*\*\*  
CREATE REST SESSION** start rest session  
**GET AUDIO SOUND** get audio sound *5***CLEAN UP** Reset Settings

Implementation:

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

**Step2**: Create generic.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 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.