Hi Team,

I have developed the scripts using PyCharm IDE,python,Robotframework and selenium.This is a very simple test suit that I prepared during limited timeframe and please go through this document if you face any difficulty in understanding. :)

Versions:

Python version : Python 3.11.1

Robot version :Robot Framework 6.0.2 (Python 3.11.1 on win32)

pip version: pip 22.3.1

Selenium:4.8.0

Note:I have highlighted my comments in blue color.

USER STORIES

==

(1) As the Clerk, I should be able to insert a single record of

working class hero into database via an API

AC1: Single record of a working class hero should consist of Natural Id

(natid), Name, Gender, Birthday, Salary and Tax paid

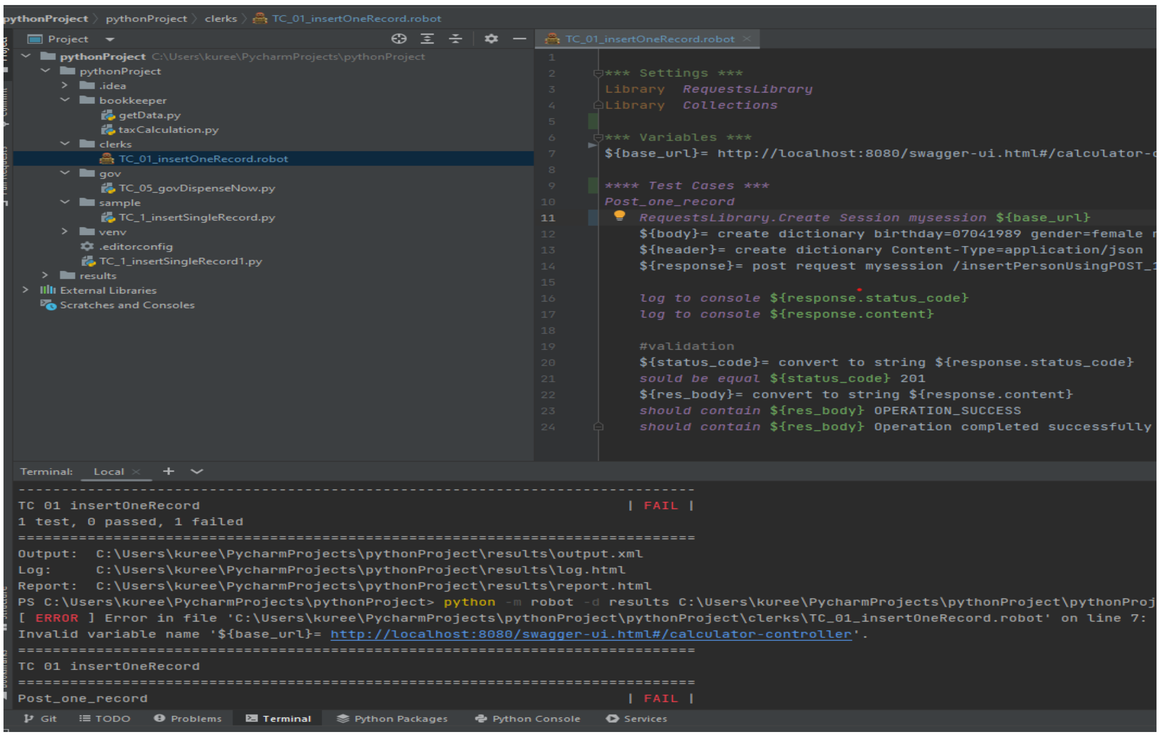
For this testcase I used robot framework and sent POST request to url:http://localhost:8080/swagger-ui.html#/calculator-controller/insertPersonUsingPOST_1.

Please refer TC01_AC01_insertOneRecord.robot file in the tst suit.

Installed robot class in my system and added pluggins in pyCharm.You can use below cmd to run the test.

```
python -m robot -d results
C:\Users\kuree\PycharmProjects\pythonProject\pythonProject\clerks\TC_01_insertOneRecord.robo
t
```

IDE snapshot:



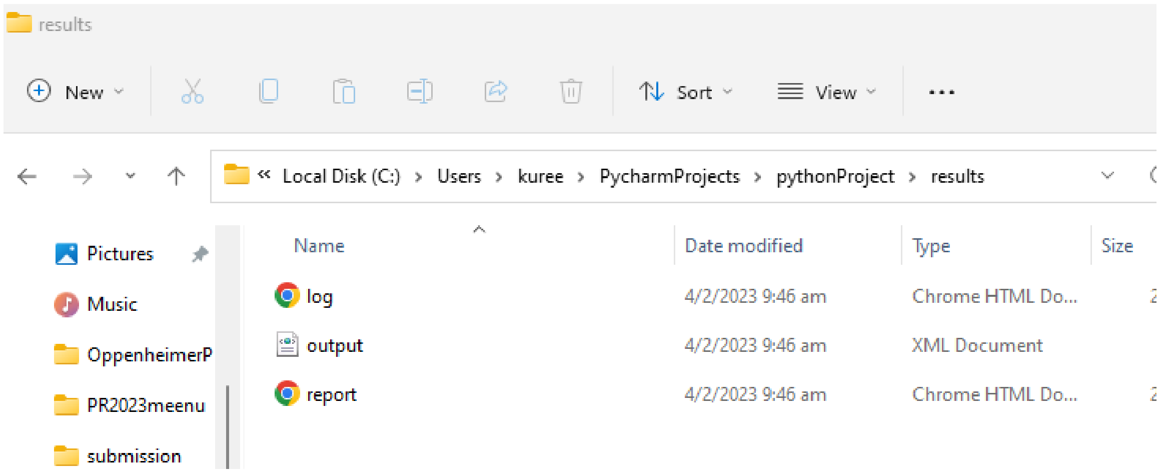
```
*** Settings ***
Library  RequestsLibrary
Library  Collections

*** Variables ***
${base_url}= http://localhost:8080/swagger-ui.html#/calculator-c

**** Test Cases ***
Post_one_record
    RequestsLibrary.Create Session mysession ${base_url}
    ${body}= create dictionary birthday=07041989 gender=female r
    ${header}= create dictionary Content-Type=application/json
    ${response}= post request mysession /insertPersonUsingPOST_1

    log to console ${response.status_code}
    log to console ${response.content}

    #validation
    ${status_code}= convert to string ${response.status_code}
    sould be equal ${status_code} 201
    ${res_body}= convert to string ${response.content}
    should contain ${res_body} OPERATION_SUCCESS
    should contain ${res_body} Operation completed successfully
```

```
------------------------------------------------------------------------------
TC 01 insertOneRecord                                                 | FAIL |
1 test, 0 passed, 1 failed
==============================================================================
Output:  C:\Users\kuree\PycharmProjects\pythonProject\results\output.xml
Log:     C:\Users\kuree\PycharmProjects\pythonProject\results\log.html
Report:  C:\Users\kuree\PycharmProjects\pythonProject\results\report.html
PS C:\Users\kuree\PycharmProjects\pythonProject> python -m robot -d results C:\Users\kuree\PycharmProjects\pythonProject\pythonProj
[ ERROR ] Error in file 'C:\Users\kuree\PycharmProjects\pythonProject\pythonProject\clerks\TC_01_insertOneRecord.robot' on line 7:
Invalid variable name '${base_url}= http://localhost:8080/swagger-ui.html#/calculator-controller'.
==============================================================================
TC 01 insertOneRecord
==============================================================================
Post_one_record                                                       | FAIL |
```

How to see the Report:

Once the test is completed its exeution, go to your project folder and open 'results' folder.

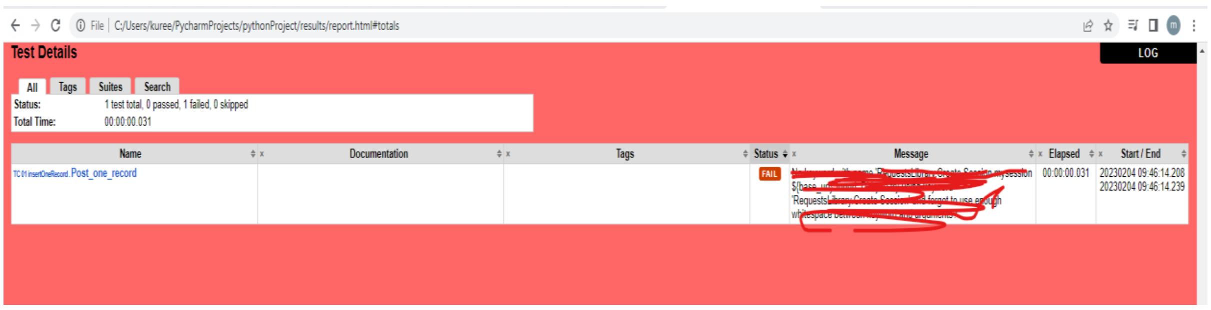
Click on 'report.html' page and see the report.


results
New
Sort
View
...
Local Disk (C:) › Users › kuree › PycharmProjects › pythonProject › results
Pictures
Music
OppenheimerP
PR2023meenu
submission

Name	Date modified	Type	Size
log	4/2/2023 9:46 am	Chrome HTML Do...	2
output	4/2/2023 9:46 am	XML Document	
report	4/2/2023 9:46 am	Chrome HTML Do...	2

HTML Report as shown below with details.

Below is the sample report.


File | C:/Users/kuree/PycharmProjects/pythonProject/results/report.html#totals
LOG
All
Tags
Suites
Search

Test Details

Status:	1 test total, 0 passed, 1 failed, 0 skipped
Total Time:	00:00:00.031

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
TC 01 insertOneRecord.Post_one_record			FAIL	[redacted]	00:00:00.031	20230204 09:46:14.208 20230204 09:46:14.239

(2) As the Clerk, I should be able to insert more than one working

class hero into database via an API

AC1: Enhancement of (1), with the ability to insert a list

For this testcase I used json file and python to send POST request to url:http://localhost:8080/swagger-ui.htinsertPersonUsingPOSTml#/calculator-controller/

Please see the testcase TC02_AC01_insertListOfRecords.robot.

(3) As the Clerk, I should be able to upload a csv file to a portal so

that I can populate the database from a UI

AC1: First row of the csv file must be natid, name, gender, salary,

birthday, tax

AC1 ,AC02 and AC3 clubbed together.

please refer TC02_AC03_uploadCSV.robot

AC2: Subsequent rows of csv are the relevant details of each working

class hero

For Ac1 and AC2,I created a csv file as given above and saved in local folder.

please refer TC03_AC03_uploadCSV.robot

AC3: A simple button that allows me to upload a file on my pc to the

portal

Functionality is not working in the application.So I completed the script in python for uploading csv file.

For this testcase I used selenium+python to upload csv file to the application.

please refer TC03_AC03_uploadCSV.robot

(4) As the Bookkeeper, I should be able to query the amount of tax

relief for each person in the database so that I can report the

figures to my Bookkeeping Manager

AC1: a GET endpoint which returns a list consist of natid, tax relief

amount and name

To gte these details I used robot framework+python to send GET request.

Used robot class and send GetT request to "http://localhost:8080/calculator/taxRelief"

saved the json file and verified the response data.

Please find the testcase TC04_AC01_GETdata&verify.robot file in test suit.

AC2: natid field must be masked from the 5th character onwards with

dollar sign '$'

Please find the testcase TC04_AC02_verifyNatidMaskedwith$.py file in test suit.

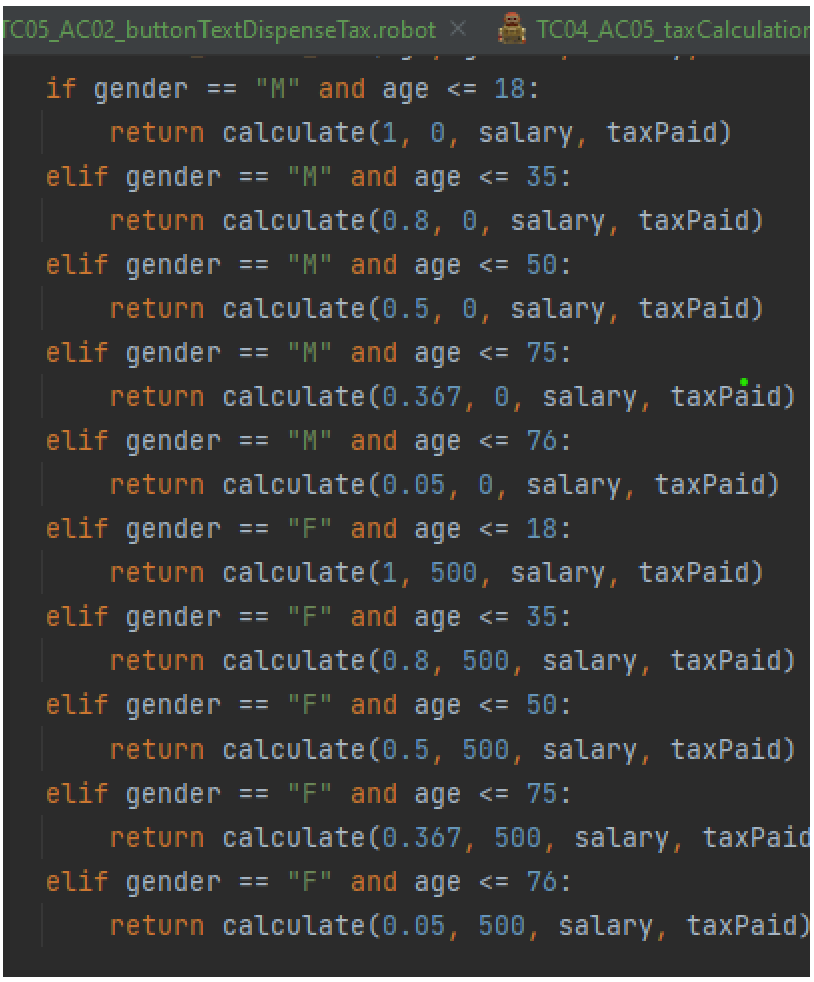
Here I used selenium+python to verify if the column of 'natid' in the table contains '$' symbol from 5th charecter.

AC3: computation of the tax relief is using the formula as described:

For this testcase I used python code to do the calculation and get data from csv.

Library used is panda.

Please refer testcase TC04_AC03_calculateTax.py



```
    if gender == "M" and age <= 18:
        return calculate(1, 0, salary, taxPaid)
    elif gender == "M" and age <= 35:
        return calculate(0.8, 0, salary, taxPaid)
    elif gender == "M" and age <= 50:
        return calculate(0.5, 0, salary, taxPaid)
    elif gender == "M" and age <= 75:
        return calculate(0.367, 0, salary, taxPaid)
    elif gender == "M" and age <= 76:
        return calculate(0.05, 0, salary, taxPaid)
    elif gender == "F" and age <= 18:
        return calculate(1, 500, salary, taxPaid)
    elif gender == "F" and age <= 35:
        return calculate(0.8, 500, salary, taxPaid)
    elif gender == "F" and age <= 50:
        return calculate(0.5, 500, salary, taxPaid)
    elif gender == "F" and age <= 75:
        return calculate(0.367, 500, salary, taxPaid
    elif gender == "F" and age <= 76:
        return calculate(0.05, 500, salary, taxPaid)
```

*AC - Acceptance Criteria

AC4: After calculating the tax relief amount, it should be subjected to

normal rounding rule to remove any decimal places

Please refer testcase TC04_AC04_normalRoundingTaxRelief.py

Applied builtin function 'round(amount)' in python.

AC5: If the calculated tax relief amount after subjecting to normal

rounding rule is more than 0.00 but less than 50.00, the final tax

relief amount should be 50.00

Please refer testcase TC04_AC05_taxCalculation0-50.py

Applied if else condition.

AC6: If the calculated tax relief amount before applying the normal

rounding rule gives a value with more than 2 decimal places, it should

be truncated at the second decimal place and then subjected to normal

rounding rule

For this testcase I used function 'format(taxRel,.2f)' function in python.

Please see the test case TC04_AC06_taxReliefRoundingafterRemoveDecimal.py

(5) As the Governor, I should be able to see a button on the screen so

that I can dispense tax relief for my working class heroes

AC1: The button on the screen must be red-colored

Please see the testcase TC05_AC01_buttonColorDispenseTax.py

AC2: The text on the button must be exactly “Dispense Now”

Please see the testcase TC05_AC02_buttonTextDispenseTax.py

AC3: After clicking on the button, it should direct me to a page with a

text that says “Cash dispensed”

For this testcase I used selenium wedriver with python to verify UI elements and its attributes.

Please see the testcase TC05_AC03_clickButtonDispenseTax.py

TASK

As the Quality Engineer for this project, you are to come up with the testing

stramust be in Python and recent version of Robot Framework. Do setup the

necessary framework onto your machine so that you can demo them during the

interview.

Yes.done.Thank you.

tegy on how to verify correctness and define quality for this system. Your

code Please commit your code to a public repository. Do send us the url once you're

done.

I have developed this small test suit in pyCharm and pushed the code to github.The url is as below.

https://github.com/meenujoseph123/pythonProject

● You may experience error(s) when testing the application, please note

down the error(s) and your approach as QE to resolve the error(s) if

any.

1.csv file upload functionality is not working in the application.

We can choose csv file from the explorer window but not selected or displayed on the screen.

2.'Refresh Tax Relief Table' button is not responding.The button is clickable but no functionality is incorporated.

My understanding is there should be a table in UI when clicking on 'Refresh Tax Relief' button, the latest details must be displayed on the table.

3.No ways to identify the login person or his role.

TIPS

● How do you test for functional requirements

First I tested manually and automated after that.

I read all the instructions and tried functionalities one by one by writing testcases manually first.

Verified each and every testable features and developed scripts.

● How do you test for non-functional requirements

Basic UI standards and userfrindliness we test manually.

Performance can test using jmeter,diffrent chrome extensions we can do.

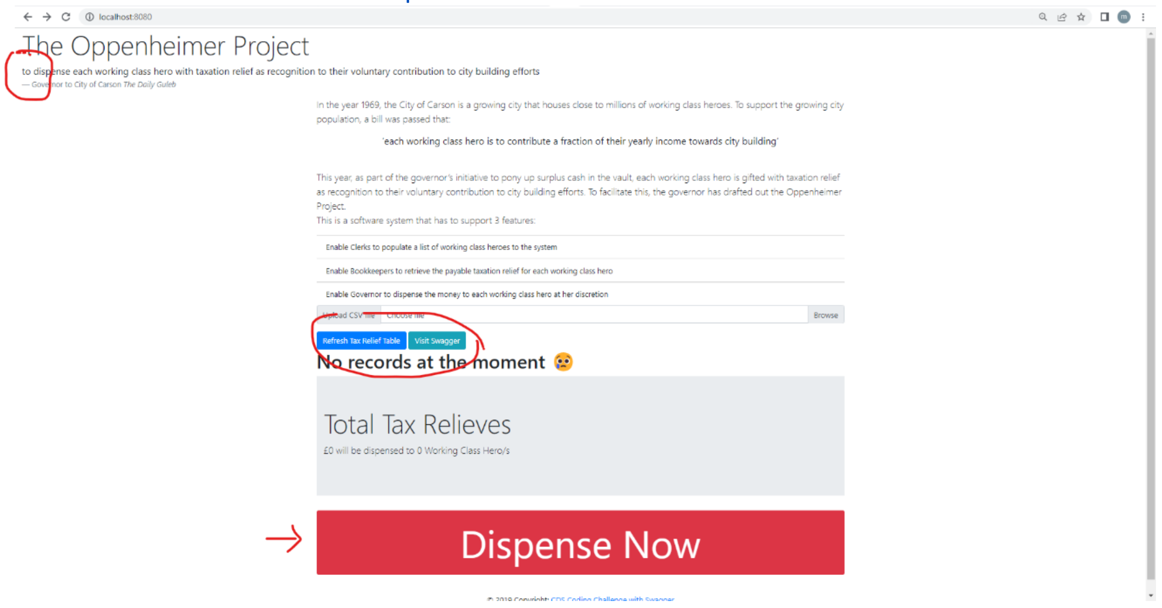
1.'Dispense Button size is large-(negative case)

2.Button alignments would be good if it comed right aligned.(negative case).can be more userfriendly.

3.Can add 'Cancel' buttons for not dispensing cases.

5.Project heading can be center aligned and in bold.

6.All sentences must start with capital letters.


localhost:8080

The Oppenheimer Project

to dispense each working class hero with taxation relief as recognition to their voluntary contribution to city building efforts

— Governor to City of Carson *The Daily Guleb*

In the year 1969, the City of Carson is a growing city that houses close to millions of working class heroes. To support the growing city population, a bill was passed that:

'each working class hero is to contribute a fraction of their yearly income towards city building'

This year, as part of the governor's initiative to pony up surplus cash in the vault, each working class hero is gifted with taxation relief as recognition to their voluntary contribution to city building efforts. To facilitate this, the governor has drafted out the Oppenheimer Project.

This is a software system that has to support 3 features:

Enable Clerks to populate a list of working class heroes to the system

Enable Bookkeepers to retrieve the payable taxation relief for each working class hero

Enable Governor to dispense the money to each working class hero at her discretion

Upload CSV file | Choose file | Browse

Refresh Tax Relief Table | Visit Swagger

No records at the moment 😢

Total Tax Relieves

£0 will be dispensed to 0 Working Class Hero/s

Dispense Now

© 2019 Copyright: CDS Coding Challenge with Swagger

7.It would be better if we can ass a cancel button to remove the selected csv file to remove from selection.

8.Verified messages.But not displayed any error message or success message except "Tax Dispensed".

● What are the test cases you've covered

1.Upload the details in to the system-failed

2.Click on 'Visit Swagger' button and verify if the system navigates to the swagger page-pass

3.Click on 'Refresh Tax Relief Table' and verify the details in table is updated-failed

4.Verify if there is no data entered , application collapse or not-pass

5.Click on 'Dispense' button and verify if the user can move to a new page with 'Tax Dispensed' message-pass

● A short Readme on how to run these tools will definitely put a smile on

our face =)

sure.Hope this document helps you.

- Feel free to add any other features which you feel necessary

1.Login feature is missing and can Implement role based access to the functionalities as explained in the scenarios given.

2. UI can be little more attractive and the button size is large and alignments are not as per the standards.

3.Tax relief table that missing can be added with filter functionality and a hyper link to open to the more details page corresponding to each heroes.

4.A tax calculator and a calendar can be add in UI(webservices can use)

Print functionality.

5.Cancel/Reject/confirm/Ok buttons(and alert popups) can add to make application more userfriendly.

6.Add a button to bring back the user from "Tax Dispersed" page.When click on "Go Back to Home Page" , the user must be able to come back to Oppenheimen project home page where "Dispense Now" button available.

7.swagger page is not need to be available to the front end user since it is application programming interface to database.

8.Could have been better if the application display error messages and success messages when requires.

9.For upload functionality, we can add cancel button and display an error message if wrong file is selected.