THE OPPENHEIMER PROJECT

BACKGROUND

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

^: also known as employees

Below are the user stories implemented:

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

(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

(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

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

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

(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

AC2: natid field must be masked from the 5th character onwards with dollar sign '$'

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

Tax Relief = ((salary - tax paid) * *variable*) + *gender bonus*			
where *variable* is determined by			
	Age	*variable*	
	age at most 18	1	
	age at most 35	0.8	
	age at most 50	0.5	
	age at most 75	0.367	
	age at least 76	0.05	
where *gender bonus* is determined by			
	Gender	*gender bonus*	
	M	0	
	F	500	

*AC - Acceptance Criteria

AC4: After calculating the tax relief amount, it should be subjected to normal rounding rule to remove any decimal places

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

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

(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

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

AC3: After clicking on the button, it should direct me to a page with a text that says “Cash dispensed”

TASK

As the Quality Engineer for this project, you are to come up with the testing strategy on how to verify correctness and define quality for this system. **Your code must 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.

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

You may locate the app and the instructions to run the app here:

https://bit.ly/3RGTBrV

Good luck and enjoy!

Note

- The date format for the birthday is: DDMMYYYY.
- 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.

TIPS

- How do you test for functional requirements
- How do you test for non-functional requirements
- What are the test cases you've covered
- A short Readme on how to run these tools will definitely put a smile on our face =)
- Feel free to add any other features which you feel necessary