Goal is to design and develop automate test using ATDD(robot framework) with the given user stories and test strategy to verify correctness and quality of the system.

Testing Strategy/Approach:

System Integration Testing : Test the system as whole by simulating usage of the application. (Ex. This includes file upload functionality, testing UI elements such as Dispense Now, Cash Dispensed and end-to-end flow of inserting records calculating and dispensing tax-relief)

API Testing : To test the API endpoints such as used to insert records and query the tax relief information. This can include testing the input validation, testing the correct computation of the tax relief formula, and testing the correct response of the API.

Acceptance Testing : Test the system against the acceptance criteria outlined in the user stories to ensure that it meets all of the requirements

Exploratory Testing : Test the system where the tester actively explores the application, without following a specific test plan or script. It allows the tester to look for new and creative ways of using the application, and to find issues that might be missed by automated tests.

High Level test scenarios were evolved

Able to drive more testcases/scenarios based on user story given here some samples tests used forAutomation

1. Validate that a single record of a working class hero can be inserted via the API by sending a POST request with the correct data format and checking the response for success.
2. Validate that multiple records of working class heroes can be inserted via the API by sending a POST request with a list of heroes and checking the response for success.
3. Validate that a CSV file can be uploaded to the portal and the data is correctly parsed and inserted by uploading a sample csv file and checking the contents of the database.
4. Validate the date format for birthday via both in individual request data and csv file upload.
5. Validate that the tax relief calculation is correct by comparing the output of the GET endpoint with a manual calculation using the formula provided in the acceptance criteria.
6. Validate that the natid field is correctly masked with dollar signs(‘$’) by checking the output of the GET endpoint.
7. Validate that the tax relief amount is correctly rounded and truncated by comparing the output of the GET endpoint with manual calculations using the rounding and truncation rules provided in the acceptance criteria.
8. Validate Json schema for tax relief GET endpoint.
9. Validate that the "Dispense Now" button is displayed with the correct color and text by checking the HTML of the page.
10. Validate that clicking on the "Dispense Now" button directs the user to the correct page by clicking the button and checking the URL of the new page.

Automation Approach:

1. **Design Pattern** used for sample test is **Page object model**.
2. **Robot Framework** used for designing the automation framework
   Used the Robot Framework, to test the System and create some functional and acceptance tests, leveraging the power of the framework to create test cases that use the keywords to interact with the system.
3. To Run and See Results, follow the quick start given below.

Quick Start (follow the steps)

1. Get latest **python3.10** and above with **pip** and **git** installed [**must]
2. Clone git folder '**QA_Assignment.git'** to local machine
3. Root folder should be '**QA_ Assignment >**' – All project files/folders parked here.
   a. Folders with Files : Resources, Results, Tests, Utils
   b. Other Files : requirements.txt, README.md

Once done with above then below steps to continue in command line terminal

1. **pip install virtualenv**
2. To isolate virtual environment, Goto '**QA_ Assignment** > Type Command: **'python -m venv qa_assess_tests**'
3. To get isolated environment, Go to folder '**qa_assess_tests**>**Scripts'** and Type: '**activate'**
4. Go back to root folder '**QA_ Assignment**
5. To install all necessary packages, need to run the sample tests, Type: '**pip install -r requirements.txt'**
6. Check test web app running locally (http://localhost:8080)

Run the test:

1. Type to run all tests: '**robot --outputdir=Results/ Tests\suiteTest.robot**'
2. Type to run only api tests: '**robot --outputdir=Results/ --include=api Tests\suiteTest.robot**'
3. Type to run only ui tests: '**robot --outputdir=Results/ --include=ui Tests\suiteTest.robot**'
4. To check reports, Go to '**/Results/report.html**'