THIS IS! DMG

Automation Framework

```
Author: Vishnu M J
Language Used: Python
Python Version: >3.0
Framework: Pytest with BDD (Gherkin)
```

Pytest-bdd solution has been used for creation of Automation Framework so as to use Cucumber+Gherkin feature files and to use full functionality of pytest. The focus has been to reduce hard-coding and improve code reusability. The Gherkin steps have been reused multiple times for the above purpose.

Pytest Test Automation Framework

Website: https://automationpractice.com/index.php

The Test Automation Suite will be having below folders.

1. Features:

- Feature files will be created based on each feature. The different scenarios in the particular feature will be listed in the same feature file.

2. Source:

- Source folder will be used to list the modules used to implement the source files for corresponding feature files. Emphasis will be kept to reuse the codes wherever possible.

3. Pages:

- This folder will be used to save the locators for each page in case of Front End Automation. Each file will be used to save the locators belonging to each page in the UI.

4. Drivers :

- To create and initialize the driver object. Implemented drivers for Chrome/ firefox and edge. Change the line browser="chrome" for different browsers

5. Utils

- Folder used to create and save custom wait or to import any wait libraries.

6. Config

- Folder used to store the Script data and other environment details.

The file is hosted in the repository https://github.com/vishnumj005/dmg-BDD.git

Test Execution

Run the following to execute the test cases using the chromedriver and selenium installed in the local

- Clone or download the project.
- Go to project root in CLI.
- Install selenium webdriver and the browser will be automatically installed once the script is run
- Run the following
    - pip install requirements.txt
    - In windows systems, please make sure to add the above to Path.
    - Now we just need to run the test cases using the below command in CLI or run the test cases in any python supported IDE by setting run configuration as pytest.

      ```
      pytest -s Source/test_0002_Login.py --alluredir= <path to report>
      pytest -s Source/test_0001_Registration.py --alluredir= <path to report>
      pytest -s Source/ --alluredir= <path to report>
      ```

    - Afte executing the script run the below command to generate allure html report.

      ```
         allure serve <path to report>
      ```

      This command will generate the HTML report and open report in default browser.

    - Execute script using tags

    -
      ```
        pytest -m login --alluredir=<path to report>

        pytest -m registration --alluredir=<path to report>
      ```

Screen Recording

```
    https://app.usebubbles.com/ma2jtGY2X2aQun7A7ecGiX/1-may-2022-comments-with-bubbles
```