Project Title: Selenium Automation Testing for OrangeHRM

This project demonstrates how to automate testing for the OrangeHRM web application using Selenium WebDriver with Python. The project includes test cases for user login, employee addition, editing, and deletion.

Tools & Technologies

1. Programming Language: Python
2. Testing Framework: pytest
3. Web Automation Tool: Selenium WebDriver
4. Web Browser: Google Chrome
5. WebDriver: ChromeDriver
6. Code Editor/IDE: PyCharm.

System Requirements

1. Operating System: Windows/Mac/Linux
2. Python Version: Python 3.x
3. Selenium Version: Selenium 4
4. pytest Version: pytest 6
5. Chrome Version: Chrome (latest stable version)
6. ChromeDriver: Compatible version of ChromeDriver for your installed Chrome browser.

Hardware Requirements

Minimum RAM: 8 GB (recommended for running tests efficiently)

CPU: Intel i5 or equivalent (for optimal performance during test executions)

Project Structure

- SeleniumOrangeHRM/
- login_page.py
- pim_page.py
- test_orange_hrm.py

CODE DETAILS

1. login_page.py

This file contains the LoginPage class that manages user login functionalities.

```
# login_page.py

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

class LoginPage:
    def __init__(self, driver):
        self.driver = driver
        self.url = "https://opensource-
demo.orangehrmlive.com/web/index.php/auth/login"
        self.selectors = {
            "username": (By.NAME, "username"),
            "password": (By.NAME, "password"),
            "login_button": (By.XPATH, '//button[@type="submit"]'),
            "login_error": (By.XPATH, '//div[@class="oxd-alert oxd-alert--
error oxd-alert--dismissible"]')
        }

    def browse(self):
        self.driver.get(self.url)
        self.driver.maximize_window()

    def wait_and_find(self, by, value):
        return WebDriverWait(self.driver,
10).until(EC.presence_of_element_located((by, value)))


    def login(self, username="Admin", password="admin123"):
        self.wait_and_find(*self.selectors["username"]).send_keys(username)
        self.wait_and_find(*self.selectors["password"]).send_keys(password)
        self.wait_and_find(*self.selectors["login_button"]).click()

    def is_login_successful(self):
        """Check if the login was successful by verifying the presence of an
error message."""
        try:
            self.wait_and_find(*self.selectors["login_error"])
            return False  # Error message found, login failed
        except:
            return True  # No error message, login successful
```

2. pim_page.py

This file contains functionalities for managing employee records.

```
# pim_page.py

from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

class PIMPage:
    def __init__(self, driver):
        self.driver = driver
        self.selectors = {
            "pim_menu": (By.XPATH,
"/html/body/div/div[1]/div[1]/aside/nav/div[2]/ul/li[2]/a"),
            "add_emp": (By.XPATH, '//button[@class="oxd-button oxd-button--
medium oxd-button--secondary"]'),
            "first_name": (By.NAME, "firstName"),
            "last_name": (By.NAME, "lastName"),
            "middle_name": (By.NAME,"middleName"),
            "emp_id": (By.XPATH,
"/html/body/div/div[1]/div[2]/div[2]/div/div/form/div[1]/div[2]/div[1]/div[2]
/div/div/div[2]/input"),
            "emp_list": (By.XPATH,
"/html/body/div/div[1]/div[1]/header/div[2]/nav/ul/li[2]/a"),
            "delete_button": (By.XPATH, '//button[@class="oxd-icon-button
oxd-table-cell-action-space"]//i[@class="oxd-icon bi-trash"]'),
            "emp_name_search": (By.XPATH, '//input[@placeholder="Type for
hints..."]'),
            "confirm_delete": (By.XPATH, '//button[@class="oxd-button oxd-
button--medium oxd-button--label-danger orangehrm-button-margin"]'),
            "edit button": (By.XPATH, '//i[@class="oxd-icon bi-pencil-
fill"]'),
            "edit_save":(By.XPATH, '//button[@class="oxd-button oxd-button--
medium oxd-button--secondary orangehrm-left-space"]'),
        }

    def wait_and_find(self, by, value):
        return WebDriverWait(self.driver,
10).until(EC.presence_of_element_located((by, value)))

    def add_employee(self, first_name, last_name, emp_id):
        self.wait_and_find(*self.selectors["pim_menu"]).click()
        self.wait_and_find(*self.selectors["add_emp"]).click()

self.wait_and_find(*self.selectors["first_name"]).send_keys(first_name)
        self.wait_and_find(*self.selectors["last_name"]).send_keys(last_name)
        self.wait_and_find(*self.selectors["emp_id"]).send_keys(emp_id)
        self.wait_and_find(By.XPATH, '//button[@type="submit"]').click()

    def edit_employee(self,first_name, middle_name):
        self.wait_and_find(*self.selectors["emp_list"]).click()

self.wait_and_find(*self.selectors["emp_name_search"]).send_keys(first_name)
        self.wait_and_find(*self.selectors["edit_button"]).click()
```

```
self.wait_and_find(*self.selectors["first_name"]).send_keys(first_name)

self.wait_and_find(*self.selectors["middle_name"]).send_keys(middle_name)
        self.wait_and_find(By.XPATH, '//button[@type="submit"]').click()




    def delete_employee(self, emp_name):
        self.wait_and_find(*self.selectors["emp_list"]).click()

self.wait_and_find(*self.selectors["emp_name_search"]).send_keys(emp_name)
        self.wait_and_find(*self.selectors["delete_button"]).click()
        self.wait_and_find(*self.selectors["confirm_delete"]).click()
```

3. test_orange_hrm.py

This file includes test cases using the `pytest` framework.

```
#test_orange_hrm.py

import pytest
from selenium import webdriver
from login_page import LoginPage
from pim_page import PIMPage


@pytest.fixture(scope="class")
def driver(request):
    driver = webdriver.Chrome()
    request.cls.driver = driver
    yield driver
    driver.quit()


@pytest.mark.usefixtures("driver")
class TestOrangeHrm:

    def test_invalid_login(self):
        self.login_page = LoginPage(self.driver)
        self.login_page.browse()  # Navigate to the login page

        # Use invalid credentials
        invalid_username = "InvalidUser"
        invalid_password = "InvalidPass"

        # Add a debug statement to show what's happening
        print("Attempting to login with invalid credentials...")
        self.login_page.login(invalid_username, invalid_password)
        print("invalid")

    def test_valid_login(self):
```

```
        self.login_page = LoginPage(self.driver)
        self.login_page.browse()

        valid_username = "Admin"
        valid_password = "admin123"
        self.login_page.login(valid_username, valid_password)

        assert self.login_page.is_login_successful(), "Login failed with
valid credentials."
        print("Login was successful.")
    def test_add_employee(self):
        # This test can be included if successful logins are handled
elsewhere
        self.pim_page = PIMPage(self.driver)

        # Add employee if logged in
        self.pim_page.add_employee("SyedAli", "Aseeka", "145326")
        print("successful employee addition.")

    def test_edit_employee(self):
        self.pim_page = PIMPage(self.driver)

        # Provide values for all required parameters

        self.pim_page.edit_employee(first_name="Faruk", middle_name="raja")
        print("successful employee updated")

    def test_delete_employee(self):
        # This test can also be included if necessary
        self.pim_page = PIMPage(self.driver)

        # Deletion should only proceed if a valid context is established
        self.pim_page.delete_employee("Faruk")
        print("successful employee deletion.")
```

Installation Instructions

1. Install Python from [python.org](https://www.python.org/downloads/).

2. Ensure you have the correct version of ChromeDriver installed for your version of Chrome. Place it in your system's PATH or specify the path in the code.

3. Run the tests & generate the report using the command:

```
pytest -v -s --capture=sys --html=Reports\HomePage_GUVI.html test_orange_hrm.py
```

Conclusion

This project showcases how to create automated tests for an application using Selenium and Python. The structure enables easy modifications and scalability for future test scenarios.