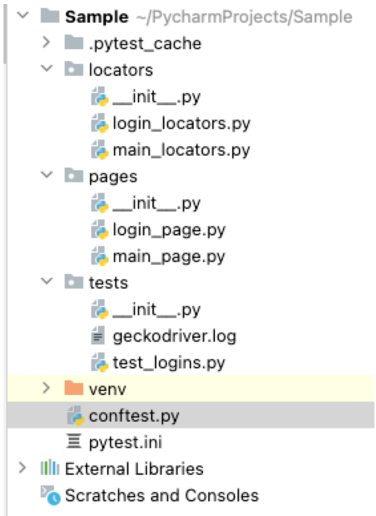

```
Sample ~/PycharmProjects/Sample
  .pytest_cache
  locators
    __init__.py
    login_locators.py
    main_locators.py
  pages
    __init__.py
    login_page.py
    main_page.py
  tests
    __init__.py
    geckodriver.log
    test_logins.py
  venv
  conftest.py
  pytest.ini
External Libraries
Scratches and Consoles
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

conftest.py

```
import pytest
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from webdriver_manager.firefox import GeckoDriverManager

URL = 'https://www.saucedemo.com/'


# @pytest.fixture(params=['firefox'], scope='function', autouse=True)
def init_driver_chrome():
    options = webdriver.ChromeOptions()
    options.add_argument("--window-size=1600,1080")
    options.headless = True
    # options.headless = False
    driver = webdriver.Chrome(
        executable_path=ChromeDriverManager().install(), options=options)
    return driver


def init_driver_firefox():
    options = webdriver.FirefoxOptions()
    options.add_argument("--window-size=1600,1080")
    options.headless = True
    # options.headless = False
    driver = webdriver.Firefox(
        executable_path=GeckoDriverManager().install(), options=options)
    return driver

# Если мы хотим, чтобы все тесты запускались на двух браузерах
```

```
@pytest.fixture(params=['chrome', 'firefox'], scope='function',
autouse=True)


# @pytest.fixture(params=['firefox'], scope='function', autouse=True)
def init_driver(request):
    driver = None
    if request.param == 'chrome':
        driver = init_driver_chrome()
    elif request.param == 'firefox':
        driver = init_driver_firefox()
    else:
        print('Please pass the correct browser name:
{}'.format(request.param))
        raise Exception('driver is not found')

    driver.get(URL)
    driver.implicitly_wait(10)
    request.cls.driver = driver
    yield driver
    driver.quit()
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

pytest.ini

```
pytest]
markers =
    smoke:
    regression:
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

locators folder

login_locators.py

```
from selenium.webdriver.common.by import By


class LoginLocators:
    input_username = (By.CSS_SELECTOR, '#user-name')
    input_password = (By.CSS_SELECTOR, '#password')
    login_btn = (By.CSS_SELECTOR, '#login-button')
    hamburger_btn = (By.CSS_SELECTOR, '#react-burger-menu-btn')
    logout_btn = (By.CSS_SELECTOR, '#logout_sidebar_link')
    title = 'Swag Labs'
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

main_locators.py

```
from selenium.webdriver.common.by import By


class MainLocators:
    product_header = (By.CSS_SELECTOR, 'span.title')
    product_title = 'Products'
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

pages folder

login_page.py

```
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.wait import WebDriverWait

from locators.login_locators import LoginLocators as ll


class LoginPage:

    def __init__(self, driver):
        self.driver = driver

    def login_title(self):
        return self.driver.title

    def action_login(self, username, password):
        self.driver.find_element(*ll.input_username).send_keys(username)
        self.driver.find_element(*ll.input_password).send_keys(password)
        self.driver.find_element(*ll.login_btn).click()

    def action_logout(self):
        WebDriverWait(self.driver,
10).until(EC.presence_of_element_located(ll.hamburger_btn))
        self.driver.find_element(*ll.hamburger_btn).click()
        WebDriverWait(self.driver,
10).until(EC.presence_of_element_located(ll.logout_btn))
        self.driver.find_element(*ll.logout_btn).click()
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

main.py

```
from locators.main_locators import MainLocators as ml


class MainPage:
    def __int__(self, driver):
        self.driver = driver

    def get_header(self):
        return self.driver.find_element(*ml.product_header).text
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢

tests folder

test_login.py

```
import random
import pytest

from pages.login_page import LoginPage
from pages.main_page import MainPage
from locators.login_locators import LoginLocators as ll
from locators.main_locators import MainLocators as ml

'''
    login_list = ['standard_user', 'locked_out_user', 'problem_user', 
'performance_glitch_user']
    password = 'secret_sauce'

    @pytest.mark.parametrize('username, password', [('standard_user', 
'secret_sauce'),
                                                    ('problem_user', 
'secret_sauce'),
                                                    ('locked_out_user', 
'secret_sauce'),

('performance_glitch_user', 'secret_sauce')
                                                    ])
'''


class TestPage:

    login_list = ['standard_user', 'performance_glitch_user']
    password = 'secret_sauce'

    @pytest.mark.parametrize('username, password', [('standard_user',
'secret_sauce'),

('performance_glitch_user', 'secret_sauce')
                                                    ])
    def test_logins(self, username, password):
        lp = LoginPage(self.driver)
        lp.action_login(username, password)
        lp.action_logout()
        assert lp.login_title() == ll.title
```

➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢ ➢