# **Task1**

Description:  
Provide the Python code based on the [api\_task.py](https://github.com/AntonLazarchik/DQE_LAB_2024_CODE_SAMPLES/blob/main/api_task.py) which will test 2 scenarios:

1. Let’s pretend that you’ve loaded the test data for the user with Id=3 and generated 10 valid posts. Verify that 10 posts were created for user 3.

Use: <https://jsonplaceholder.typicode.com/> (Resources section)

2. You have some data project on which the client used previously GCP cloud for storing the data for forecasts. Now the client wants to build some analytics around this data in AWS. So, we need to verify that the data for some specific date is presented on the source (GCP) and the target staging bucket in AWS. You need to verify in the test that both buckets for specific dates are not empty as a basic smoke test.1. Using the automatic approach (Selenium Manager or Driver Management Software) for Chrome.

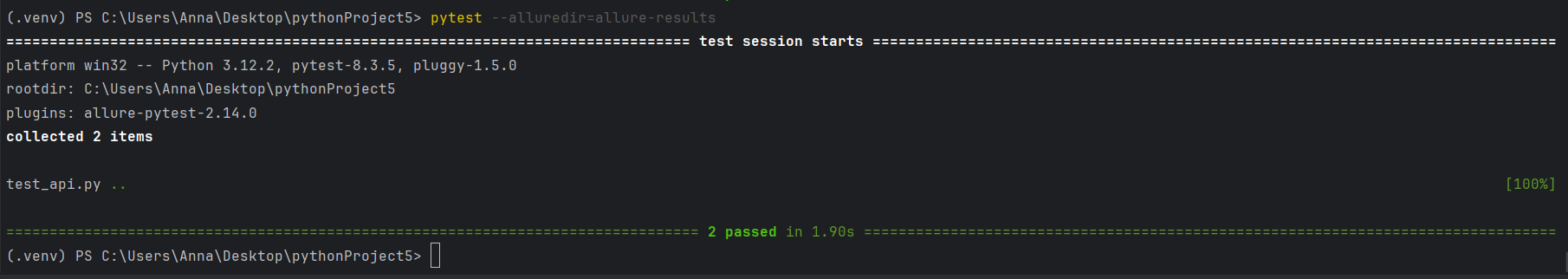
**Expected result:** Python script, Conftest,py file with fixtures, a screenshot of the allure report with 2 tests.

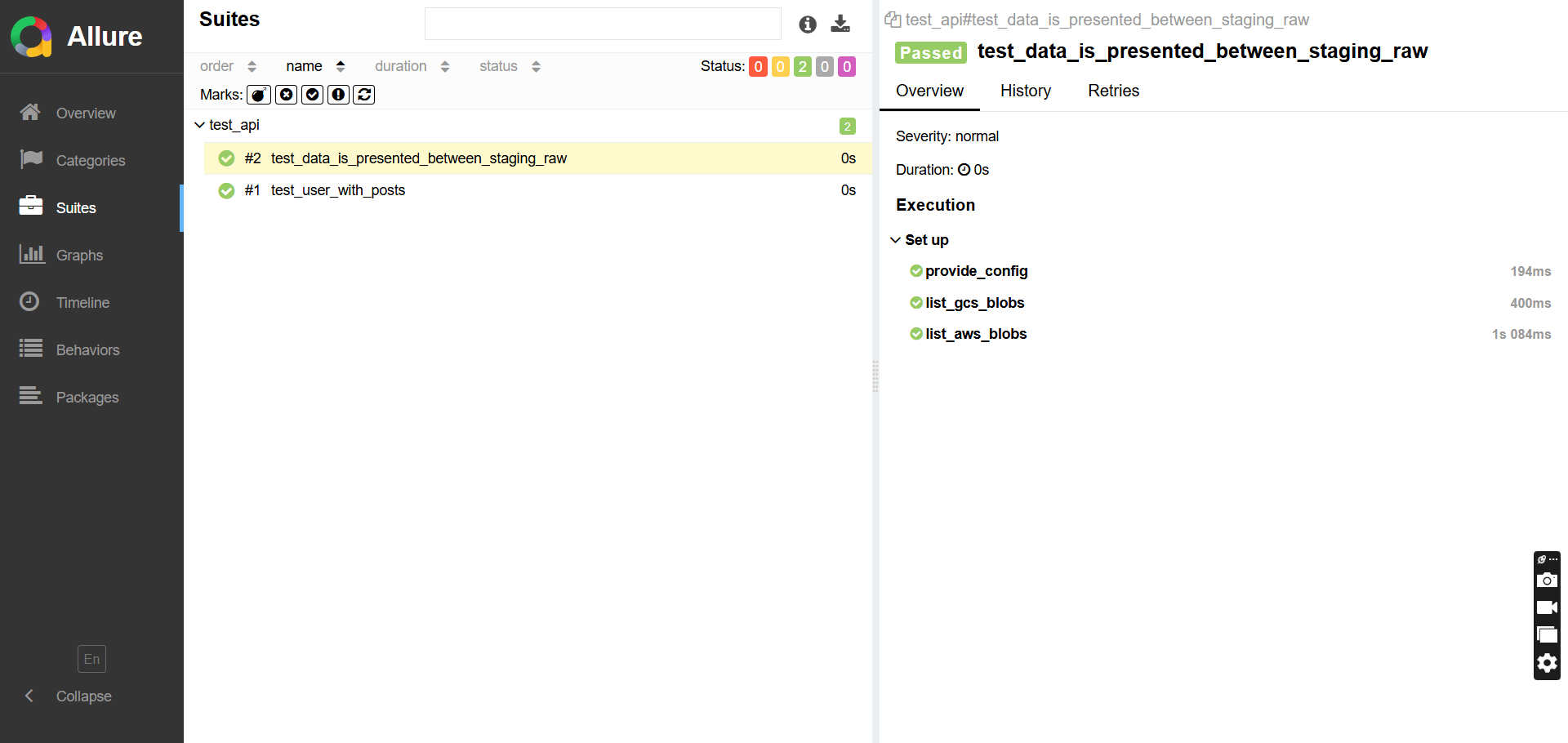
Conftest.py:

import pytest  
import boto3  
from botocore import UNSIGNED  
from botocore.config import Config  
from google.cloud import storage  
import requests  
from selenium import webdriver  
  
  
@pytest.fixture(scope='session')  
def selenium\_driver():  
 driver = webdriver.Chrome()  
 yield driver  
 driver.quit()  
  
  
@pytest.fixture(scope='function')  
def provide\_config():  
 return {  
 'prefix': '2024/01/01/KTLX/',  
 'gcp\_bucket\_name': "gcp-public-data-nexrad-l2",  
 'aws\_bucket\_name': 'noaa-nexrad-level2',  
 's3\_anon\_client': boto3.client('s3', config=Config(signature\_version=UNSIGNED)),  
 'gcp\_storage\_anon\_client': storage.Client.create\_anonymous\_client()  
 }  
  
  
@pytest.fixture(scope='function')  
def list\_gcs\_blobs(provide\_config):  
 cfg = provide\_config  
 blobs = cfg['gcp\_storage\_anon\_client'].list\_blobs(  
 cfg['gcp\_bucket\_name'],  
 prefix=cfg['prefix']  
 )  
 return [blob.name for blob in blobs]  
  
  
@pytest.fixture(scope='function')  
def list\_aws\_blobs(provide\_config):  
 cfg = provide\_config  
 resp = cfg['s3\_anon\_client'].list\_objects(  
 Bucket=cfg['aws\_bucket\_name'],  
 Prefix=cfg['prefix']  
 )  
 if 'Contents' not in resp:  
 return []  
 return [item['Key'] for item in resp['Contents']]  
  
  
@pytest.fixture(scope='function')  
def provide\_posts\_data():  
 url = "https://jsonplaceholder.typicode.com/posts"  
 params = {'userId': 3}  
 resp = requests.get(url, params=params)  
 resp.raise\_for\_status()  
 return resp.json()

test\_api.py:

def test\_user\_with\_posts(provide\_posts\_data):  
 posts = provide\_posts\_data  
 assert isinstance(posts, list), "Expected a list of posts"  
 assert len(posts) == 10, f"User 3 should have 10 posts, found {len(posts)}"  
  
  
def test\_data\_is\_presented\_between\_staging\_raw(list\_gcs\_blobs, list\_aws\_blobs):  
 gcs\_objs = list\_gcs\_blobs  
 aws\_objs = list\_aws\_blobs  
  
 assert len(gcs\_objs) > 0, "GCS bucket returned zero objects"  
 assert len(aws\_objs) > 0, "AWS S3 bucket returned zero objects"





# **Task2**

Description:

Provide the Python code which will open google.com in Chrome and Firefox browsers and print the title of the page.

1. Using the automatic approach (Selenium Manager or Driver Management Software) for Chrome.

2. Using the manual approach (PATH variable or Hard Coded Location) for Firefox.

**Expected result:** Python script

open\_browser.py:

from selenium import webdriver  
from selenium.webdriver.firefox.service import Service as FirefoxService  
  
  
def open\_with\_chrome():  
 driver = webdriver.Chrome()  
 driver.get("https://www.google.com")  
 print("Chrome Title:", driver.title)  
  
 driver.quit()  
  
  
def open\_with\_firefox(geckodriver\_path: str):  
 service = FirefoxService(executable\_path=geckodriver\_path)  
 driver = webdriver.Firefox(service=service)  
  
 driver.get("https://www.google.com")  
 print("Firefox Title:", driver.title)  
  
 driver.quit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 open\_with\_chrome()  
 gecko\_path = r"C:\Users\Anna\Desktop\pythonProject5\geckodriver.exe"  
 open\_with\_firefox(gecko\_path)

# **Task3**

Description:  
Using the attached list of screenshots and links to websites provide a list of the locators for elements in the red area. For each type of locator below prepare 2 examples. 2 locators by id, 2 locators by class name, etc. Be sure that your locator is leading to an exact single element, not a group of them.

You can provide different locators for the same element, like 1 locator by ID and 1 by name for a button element. But the maximum number of locators for the same element mustn’t be more than 2.

1. class name

Example: find\_element(By.CLASS\_NAME, "information")

2. id

Example: find\_element(By.ID, "lname")

3. name

Example: find\_element(By.NAME, "newsletter")

4. CSS selector

Example: find\_element(By.CSS\_SELECTOR, "#fname")

5. XPath

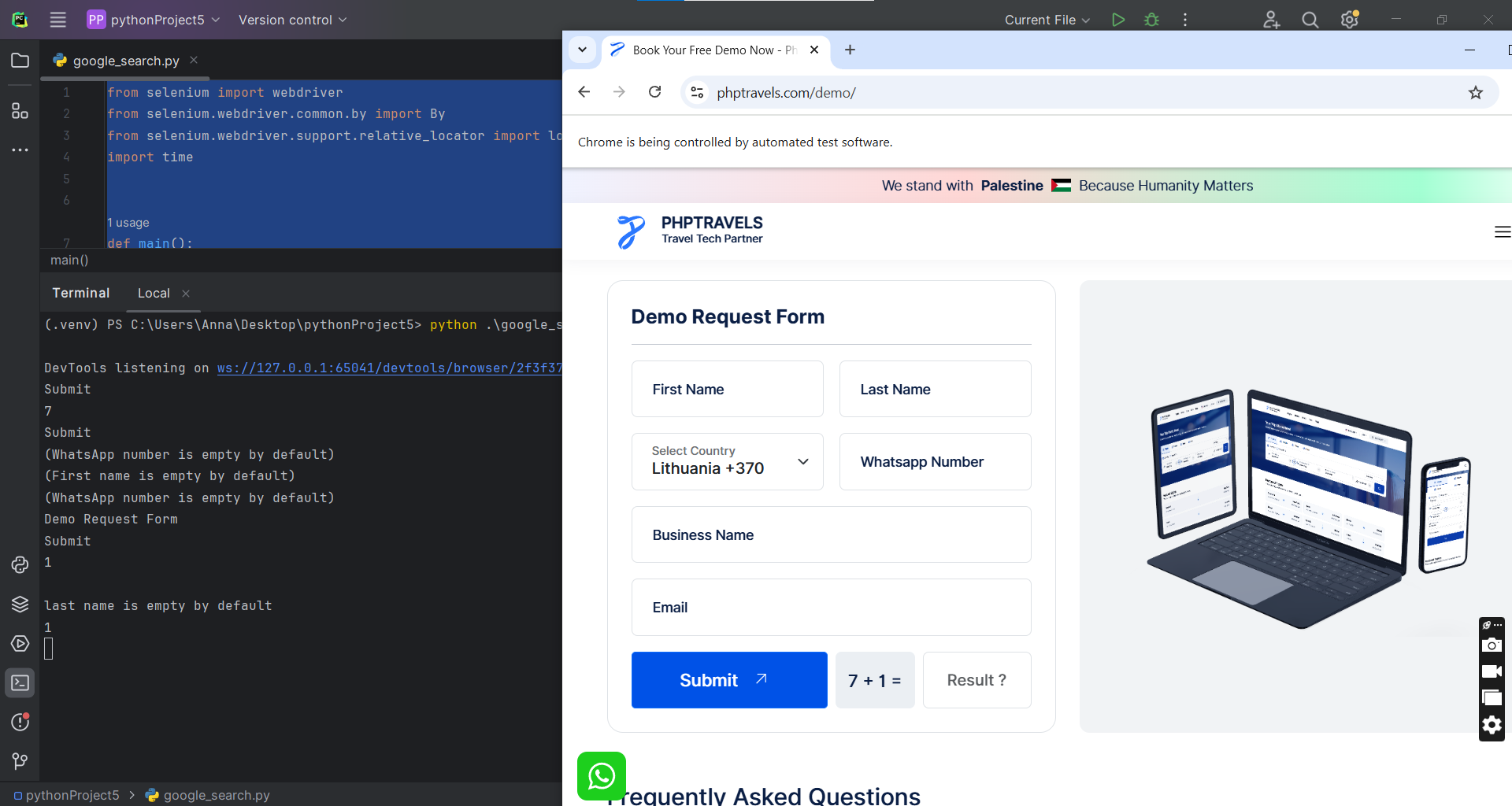
Example: find\_element(By.XPATH, "//input[@value='f']")

6\*. Relative locators

Example: locate\_with(By.TAG\_NAME, "input").above({By.ID: "password"})

**Expected result:** Python script. Screenshot with the execution results.

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.relative\_locator import locate\_with  
import time  
  
  
def main():  
 driver = webdriver.Chrome()  
 driver.implicitly\_wait(10)  
  
 driver.get("https://phptravels.com/demo/")  
  
 submit\_btn\_by\_id = driver.find\_element(By.ID, "demo")  
 print(submit\_btn\_by\_id.text)  
 number1 = driver.find\_element(By.ID, "numb1")  
 print(number1.text)  
  
 submit\_btn\_by\_class = driver.find\_element(By.CLASS\_NAME, "btn\_submit")  
 print(submit\_btn\_by\_class.text)  
 whatsapp\_number\_by\_class = driver.find\_element(By.CLASS\_NAME, "whatsapp\_number")  
 print(whatsapp\_number\_by\_class.text + "(WhatsApp number is empty by default)")  
  
 first\_name\_by\_name = driver.find\_element(By.NAME, "first\_name")  
 print(first\_name\_by\_name.text + "(First name is empty by default)")  
 whatsapp\_number\_by\_name = driver.find\_element(By.NAME, "whatsapp")  
 print(whatsapp\_number\_by\_name.text + "(WhatsApp number is empty by default)")  
  
 title\_form\_by\_css = driver.find\_element(By.CSS\_SELECTOR, "h5")  
 print(title\_form\_by\_css.text)  
 submit\_btn\_by\_css = driver.find\_element(By.CSS\_SELECTOR, ".btn-primary")  
 print(submit\_btn\_by\_css.text)  
  
 second\_number\_by\_xp = driver.find\_element(By.XPATH, "//\*[@id='numb2']")  
 print(second\_number\_by\_xp.text)  
 last\_name\_by\_xp = driver.find\_element(By.XPATH, "//\*[@id='swup']/section[1]/div/div/div["  
 "1]/div/div/div/div/div/div/div/div[1]/div[1]/div[2]/div/input")  
 print(last\_name\_by\_xp.text)  
  
 last\_name\_by\_relative = driver.find\_element(  
 locate\_with(By.TAG\_NAME, "input").to\_right\_of({By.NAME: "first\_name"})  
 )  
 print(last\_name\_by\_relative.text+"last name is empty by default")  
 num2\_by\_relative = driver.find\_element(  
 locate\_with(By.TAG\_NAME, "span").to\_right\_of({By.ID: "numb1"})  
 )  
 print(num2\_by\_relative.text)  
  
 time.sleep(2)  
 driver.quit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()



# **Task4**

Description: Prepare a Python script. Use implicit wait and explicit wait inside the script:

1. Open the google.com.
2. Specify “Selenium” as the search field.
3. Open the first link in the results.

**Expected result:** Python script. Screenshot with the execution results.

Google\_search.py:

import undetected\_chromedriver as uc  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
from selenium.common.exceptions import TimeoutException  
import time  
import random  
  
options = uc.ChromeOptions()  
options.add\_argument("--start-maximized")  
driver = uc.Chrome(options=options)  
  
driver.implicitly\_wait(10)  
  
try:  
 driver.get("https://www.google.com/?hl=en")  
 time.sleep(random.uniform(2, 3))  
  
 try:  
 consent\_button = WebDriverWait(driver, 5).until(  
 EC.element\_to\_be\_clickable((  
 By.XPATH, '//button[.//div[contains(text(), "Accept all")] or contains(text(), "I agree")]'  
 ))  
 )  
 consent\_button.click()  
 time.sleep(random.uniform(1, 2))  
 except TimeoutException:  
 print("No visible consent form")  
  
 search\_box = WebDriverWait(driver, 10).until(  
 EC.element\_to\_be\_clickable((By.NAME, "q"))  
 )  
  
 for char in "Selenium":  
 search\_box.send\_keys(char)  
 time.sleep(random.uniform(0.1, 0.3))  
  
 search\_box.send\_keys(Keys.RETURN)  
 time.sleep(random.uniform(2, 3))  
 first\_result = WebDriverWait(driver, 10).until(  
 EC.element\_to\_be\_clickable((By.CSS\_SELECTOR, "h3"))  
 )  
 first\_result.click()  
 time.sleep(random.uniform(3, 4))  
  
finally:  
 driver.quit()

