task 1:
this is confest.py code:

```
import boto3
import pytest
import requests
from botocore import UNSIGNED
from botocore.config import Config
from google.cloud import storage

@pytest.fixture(scope='function')
def provide_config():
    config = {
        '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()
    }
    return config

@pytest.fixture(scope='function')
def list_gcs_blobs(provide_config):
    config = provide_config
    blobs =
config['gcp_storage_anon_client'].list_blobs(config['gcp_bucket_name'],
prefix=config['prefix'])
    objects = [blob.name for blob in blobs]
    return objects

@pytest.fixture(scope='function')
def list_aws_blobs(provide_config):
    config = provide_config
    response =
config['s3_anon_client'].list_objects(Bucket=config['aws_bucket_name'],
Prefix=config['prefix'])
    objects = [content['Key'] for content in response.get('Contents', [])]
    return objects

@pytest.fixture(scope='function')
def provide_posts_data():
    response = requests.get('https://jsonplaceholder.typicode.com/posts',
```

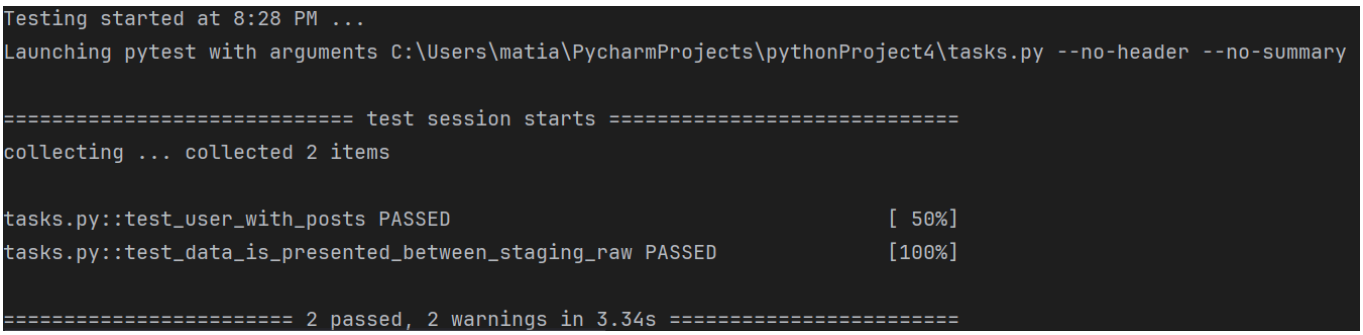
```
params={'userId': 3})
    response.raise_for_status()
    return response.json()
```

this is first class code:

```
def test_user_with_posts(provide_posts_data):
    posts = provide_posts_data
    assert len(posts) == 10, f"Expected 10 posts for user 3, but got
{len(posts)}."

def test_data_is_presented_between_staging_raw(list_gcs_blobs,
list_aws_blobs):
    assert len(list_gcs_blobs) > 0, "GCP bucket is empty for the specified
date prefix!"
    assert len(list_aws_blobs) > 0, "AWS bucket is empty for the specified
date prefix!"
```

and screenshots of the tests:



task 2:
this is the python script:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.firefox.service import Service as FirefoxService
from selenium.webdriver.common.by import By
import time

# --- PART 1: Chrome (Automatic Driver Management) ---
def open_google_in_chrome():
    driver = webdriver.Chrome()
    driver.get("https://www.google.com")
    print("Chrome Browser Title:", driver.title)
    time.sleep(2)
    driver.quit()
```

```
# --- PART 2: Firefox (Manual Driver Location) ---
def open_google_in_firefox():
    firefox_driver_path = r"C:\Users\matia\Downloads\geckodriver-v0.36.0-win-
aarch64.zip"
    service = FirefoxService(executable_path=firefox_driver_path)
    driver = webdriver.Firefox(service=service)
    driver.get("https://www.google.com")
    print("Firefox Browser Title:", driver.title)
    time.sleep(2)
    driver.quit()

if __name__ == "__main__":
    open_google_in_chrome()
    open_google_in_firefox()
```

task 3:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service as ChromeService
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException,
TimeoutException, WebDriverException


def try_find_element(driver, by, selector, wait_time=5):
    try:
        element = WebDriverWait(driver, wait_time).until(
            EC.presence_of_element_located((by, selector))
        )
        print(f"✓ Found element: {by}='{selector}'")
        return element
    except (NoSuchElementException, TimeoutException):
        print(f"✗ Element not found: {by}='{selector}'")
        return None
    except WebDriverException as e:
        print(f"! Error finding element {by}='{selector}': {e}")
        return None


def scrape_website(url, locators_dict):
    try:
```

```
        print(f"\n=== Navigating to {url} ===")
        driver.get(url)

        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.TAG_NAME, "body"))
        )

        for locator_type, locators in locators_dict.items():
            print(f"\n--- Testing {locator_type} locators ---")
            for selector in locators:
                try_find_element(driver, locator_type, selector)

    except WebDriverException as e:
        print(f"! Error navigating to {url}: {e}")


options = webdriver.ChromeOptions()
options.add_argument("--no-sandbox")
options.add_argument("--disable-dev-shm-usage")
options.add_argument("--disable-gpu")
options.add_experimental_option('excludeSwitches', ['enable-logging'])

try:
    print("Setting up Chrome driver...")
    driver =
webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()),
options=options)
    driver.maximize_window()

    driver.implicitly_wait(5)

    website1_locators = {
        By.CLASS_NAME: ["first_name", "last_name"],
        By.NAME: ["first_name", "last_name", "business_name", "email"],
        By.CSS_SELECTOR: ["input[name='first_name']",
"input[name='last_name']", ".form-control"],
        By.XPATH: ["//input[@name='first_name']",
"//input[@name='last_name']", "//button[contains(text(), 'Submit')]"]
    }
    scrape_website("https://phptravels.com/demo/", website1_locators)

    website2_locators = {
        By.CLASS_NAME: ["form-control", "form-group"],
        By.ID: ["inputFirstName", "inputLastName", "inputEmail"],
        By.NAME: ["firstname", "lastname", "email"],
        By.CSS_SELECTOR: ["#inputFirstName", "#inputLastName", ".recaptcha"],
```

```
        By.XPATH: ["//input[@id='inputFirstName']",
"//input[@id='inputLastName']"]
    }
    scrape_website("https://phptravels.org/register.php", website2_locators)

    website3_locators = {
        By.CLASS_NAME: ["post-title", "post-meta", "blog-content"],
        By.CSS_SELECTOR: [".post-title", ".post-meta", "article", ".blog-
post"],
        By.XPATH: ["//h2[contains(@class,'post-title')]/a",
"//div[contains(@class,'post-meta')]", "//article"]
    }
    scrape_website("https://phptravels.com/blog/", website3_locators)

except Exception as e:
    print(f"An error occurred: {e}")

finally:
    try:
        if 'driver' in locals():
            print("\n=== Closing browser ===")
            driver.quit()
    except Exception as e:
        print(f"Error closing browser: {e}")
```

```
=== Navigating to https://phptravels.com/demo/ ===

--- Testing class name locators ---
✓ Found element: class name='first_name'
✓ Found element: class name='last_name'

--- Testing name locators ---
✓ Found element: name='first_name'
✓ Found element: name='last_name'
✓ Found element: name='business_name'
✓ Found element: name='email'

--- Testing css selector locators ---
✓ Found element: css selector='input[name='first_name']'
✓ Found element: css selector='input[name='last_name']'
✓ Found element: css selector='.form-control'

--- Testing xpath locators ---
✓ Found element: xpath='//input[@name='first_name']'
✓ Found element: xpath='//input[@name='last_name']'
× Element not found: xpath='//button[contains(text(), 'Submit')]'
```

task 4

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service as ChromeService
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

driver = None

try:
    options = webdriver.ChromeOptions()
    options.add_argument("--start-maximized")
    driver =
webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()),
options=options)

    driver.implicitly_wait(10)

    driver.get("https://www.google.com")
    print("Opened Google")
```

```
    try:
        consent_buttons = driver.find_elements(By.XPATH,
"//button[contains(text(), 'Accept') or contains(text(), 'I agree')]")
        if len(consent_buttons) > 0:
            consent_buttons[0].click()
            print("Clicked consent button")
    except:
        pass

    search_input = driver.find_element(By.NAME, "q")
    search_input.clear()
    search_input.send_keys("Selenium")
    search_input.submit()
    print("Submitted search for 'Selenium'")

    WebDriverWait(driver, 10).until(
        EC.title_contains("Selenium")
    )
    print("Search results page loaded")

    driver.save_screenshot("search_results.png")
    print("Saved screenshot of search results as search_results.png")

    try:
        selectors = [
            "div.g h3",
            ".LC20lb",
            "h3.LC20lb",
            "#search a h3"
        ]

        for selector in selectors:
            results = driver.find_elements(By.CSS_SELECTOR, selector)
            if results:
                WebDriverWait(driver, 10).until(
                    EC.element_to_be_clickable((By.CSS_SELECTOR, selector))
                )
                results[0].click()
                print(f"Clicked first result using selector: {selector}")
                break

        time.sleep(3)

        driver.save_screenshot("result_page.png")
        print(f"Saved screenshot of result page as result_page.png")
```

```
            print(f"Successfully opened: {driver.title}")

        except Exception as e:
            print(f"Error clicking search result: {str(e)}")

    except Exception as e:
        print(f"An error occurred: {str(e)}")

    finally:
        if driver:
            driver.quit()
        print("Test completed.")
```