Software Quality Engineering (SE-309)

ASSIGNMENT 1

EXPLORATION OF SOFTWARE TESTING TOOL

(SELENIUM and POSTMAN)

Submitted by: AQSA ZAIB

Roll no : SE-21013

Date: 29/April/2024

SELENIUM:

Introduction:

Selenium is an open source testing tool that allows users to test web applications across different browsers and platforms. Selenium includes a suite of software that developers can use to automate web applications including IDE, RC, WebDriver and Selenium grid, which all serve different purposes

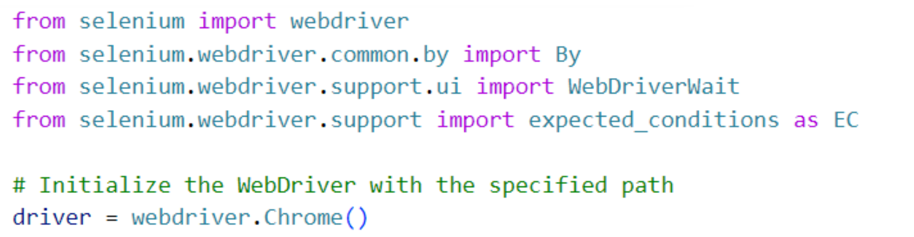
Use of selenium:

The uses of Selenium increase as you advance in the suite, and each component has advantages and disadvantages. The basic fundamentals of Selenium though revolve around browser testing, as developers and testers alike love the open source framework's ability to mimic a real user's actions in the browser.

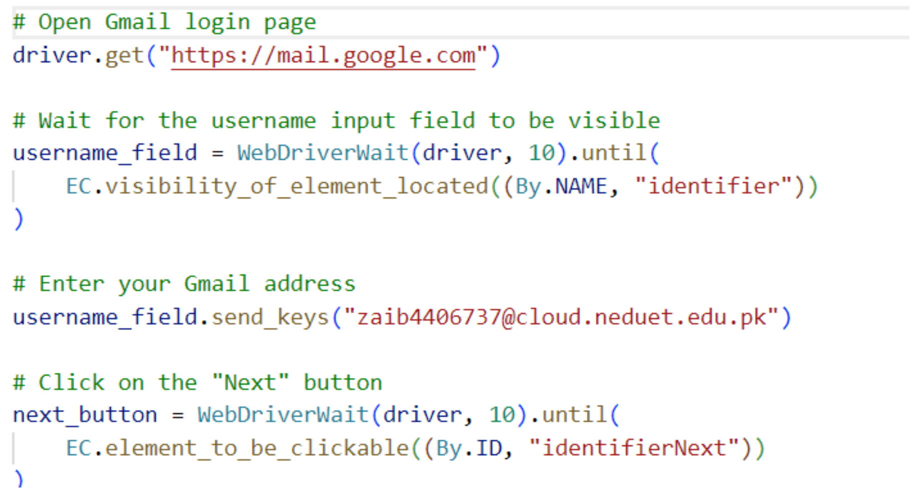
What makes Selenium so easy to learn is its ability to bind to several popular languages, letting you write in your preferred programming language including Java, Javascript, PHP, Perl, C#, Ruby, .Net, and Python.

Test Cases:

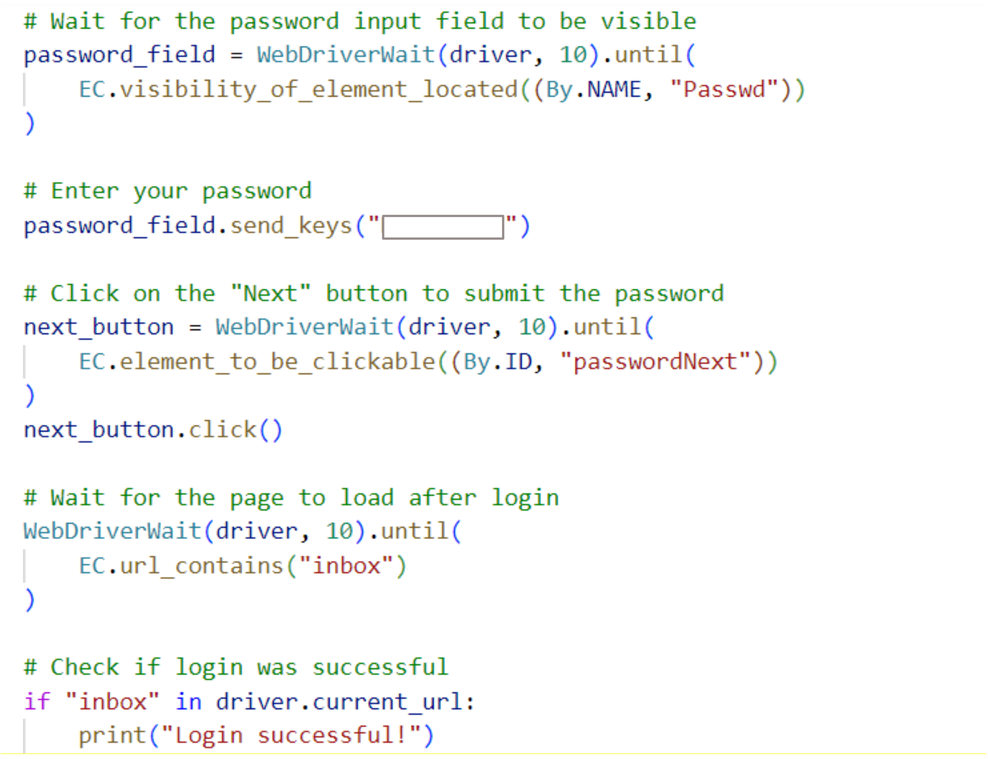
- **Gmail Login Test**
  Code:



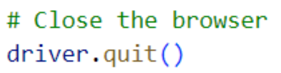
```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Initialize the WebDriver with the specified path
driver = webdriver.Chrome()
```



```
# Open Gmail login page
driver.get("https://mail.google.com")

# Wait for the username input field to be visible
username_field = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.NAME, "identifier"))
)

# Enter your Gmail address
username_field.send_keys("zaib4406737@cloud.neduet.edu.pk")

# Click on the "Next" button
next_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, "identifierNext"))
)
```

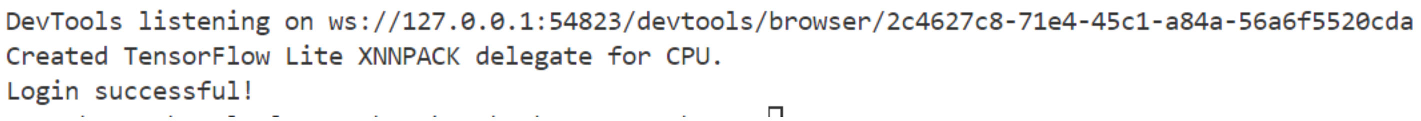


```
# Wait for the password input field to be visible
password_field = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.NAME, "Passwd"))
)

# Enter your password
password_field.send_keys("[redacted]")

# Click on the "Next" button to submit the password
next_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.ID, "passwordNext"))
)
next_button.click()

# Wait for the page to load after login
WebDriverWait(driver, 10).until(
    EC.url_contains("inbox")
)

# Check if login was successful
if "inbox" in driver.current_url:
    print("Login successful!")
```



```
# Close the browser
driver.quit()
```

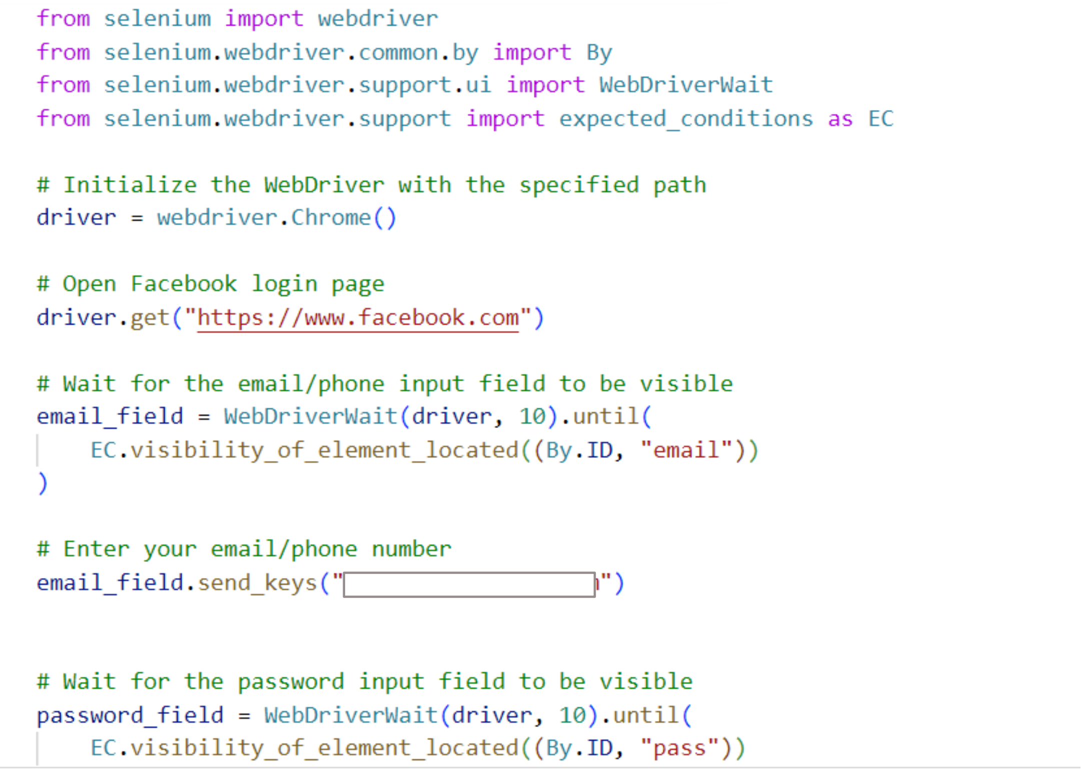
Output:



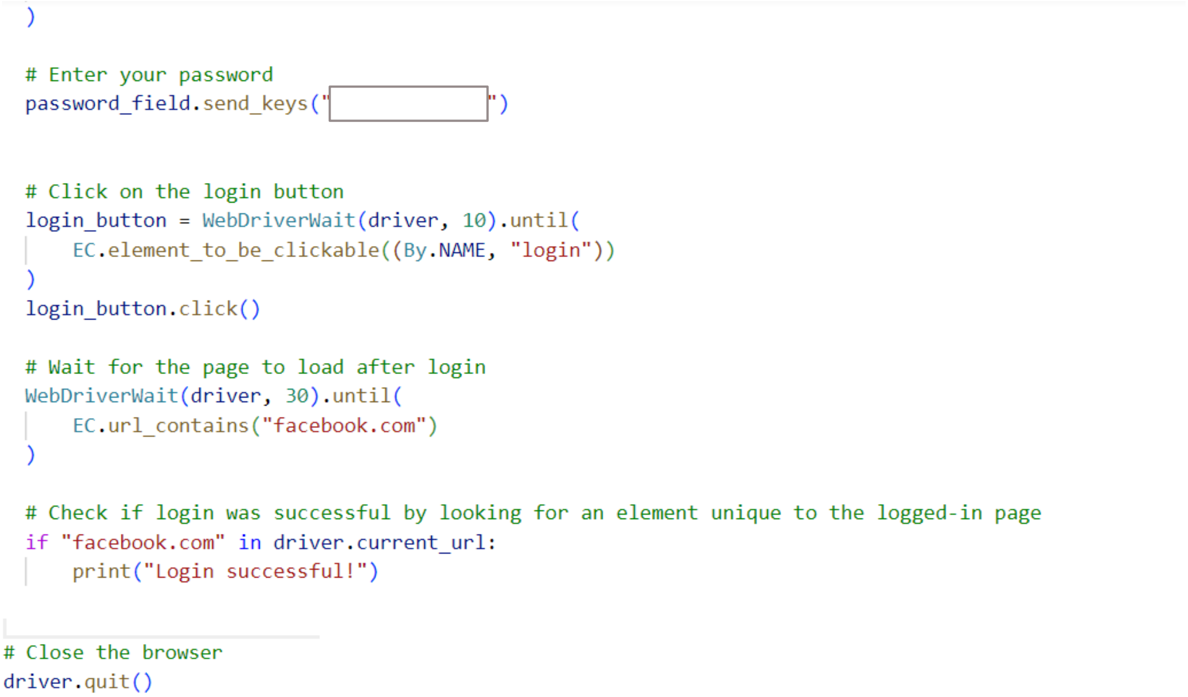
```
DevTools listening on ws://127.0.0.1:54823/devtools/browser/2c4627c8-71e4-45c1-a84a-56a6f5520cda
Created TensorFlow Lite XNNPACK delegate for CPU.
Login successful!
```

- **Facebook Login Test**

  Code:

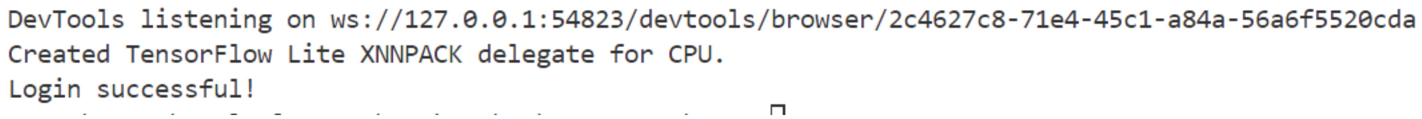


```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Initialize the WebDriver with the specified path
driver = webdriver.Chrome()

# Open Facebook login page
driver.get("https://www.facebook.com")

# Wait for the email/phone input field to be visible
email_field = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.ID, "email"))
)

# Enter your email/phone number
email_field.send_keys("[redacted]")


# Wait for the password input field to be visible
password_field = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.ID, "pass"))
```



```
    )

    # Enter your password
    password_field.send_keys("")


    # Click on the login button
    login_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.NAME, "login"))
    )
    login_button.click()

    # Wait for the page to load after login
    WebDriverWait(driver, 30).until(
        EC.url_contains("facebook.com")
    )

    # Check if login was successful by looking for an element unique to the logged-in page
    if "facebook.com" in driver.current_url:
        print("Login successful!")


# Close the browser
driver.quit()
```

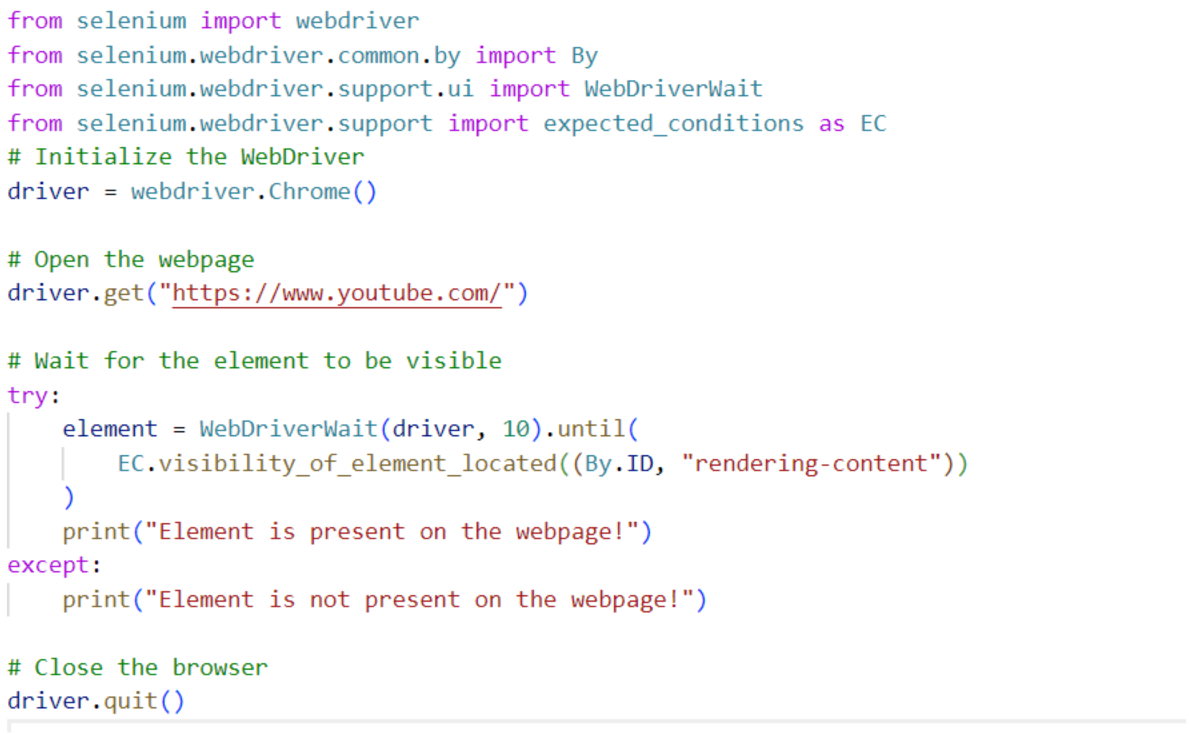
Output:



```
DevTools listening on ws://127.0.0.1:54823/devtools/browser/2c4627c8-71e4-45c1-a84a-56a6f5520cda
Created TensorFlow Lite XNNPACK delegate for CPU.
Login successful!
```

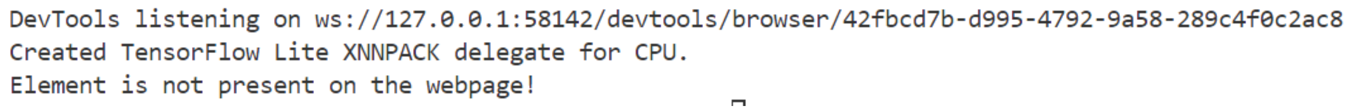
- **Presence of an element Test**

  Code:



```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
# Initialize the WebDriver
driver = webdriver.Chrome()

# Open the webpage
driver.get("https://www.youtube.com/")

# Wait for the element to be visible
try:
    element = WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.ID, "rendering-content"))
    )
    print("Element is present on the webpage!")
except:
    print("Element is not present on the webpage!")

# Close the browser
driver.quit()
```

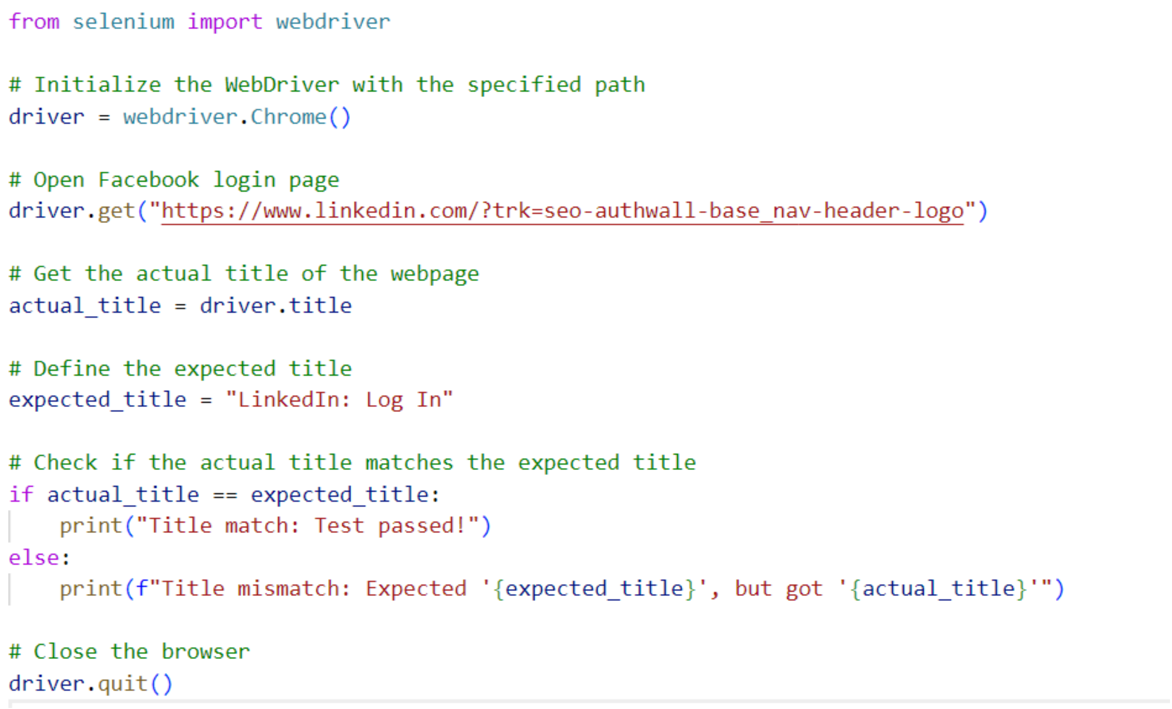
Output:



```
DevTools listening on ws://127.0.0.1:58142/devtools/browser/42fbcd7b-d995-4792-9a58-289c4f0c2ac8
Created TensorFlow Lite XNNPACK delegate for CPU.
Element is not present on the webpage!
```

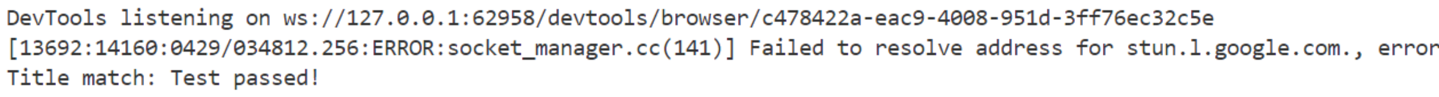
- **Actual Title Test**

  Code:



```python
from selenium import webdriver

# Initialize the WebDriver with the specified path
driver = webdriver.Chrome()

# Open Facebook login page
driver.get("https://www.linkedin.com/?trk=seo-authwall-base_nav-header-logo")

# Get the actual title of the webpage
actual_title = driver.title

# Define the expected title
expected_title = "LinkedIn: Log In"

# Check if the actual title matches the expected title
if actual_title == expected_title:
    print("Title match: Test passed!")
else:
    print(f"Title mismatch: Expected '{expected_title}', but got '{actual_title}'")

# Close the browser
driver.quit()
```

Output:



```
DevTools listening on ws://127.0.0.1:62958/devtools/browser/c478422a-eac9-4008-951d-3ff76ec32c5e
[13692:14160:0429/034812.256:ERROR:socket_manager.cc(141)] Failed to resolve address for stun.l.google.com., error
Title match: Test passed!
```

Introduction:

Selenium is an open source testing tool that allows users to test web applications across different browsers and platforms. Selenium includes a suite of software that developers can use to automate web applications including IDE, RC, WebDriver and Selenium grid, which all serve different purposes

Postman

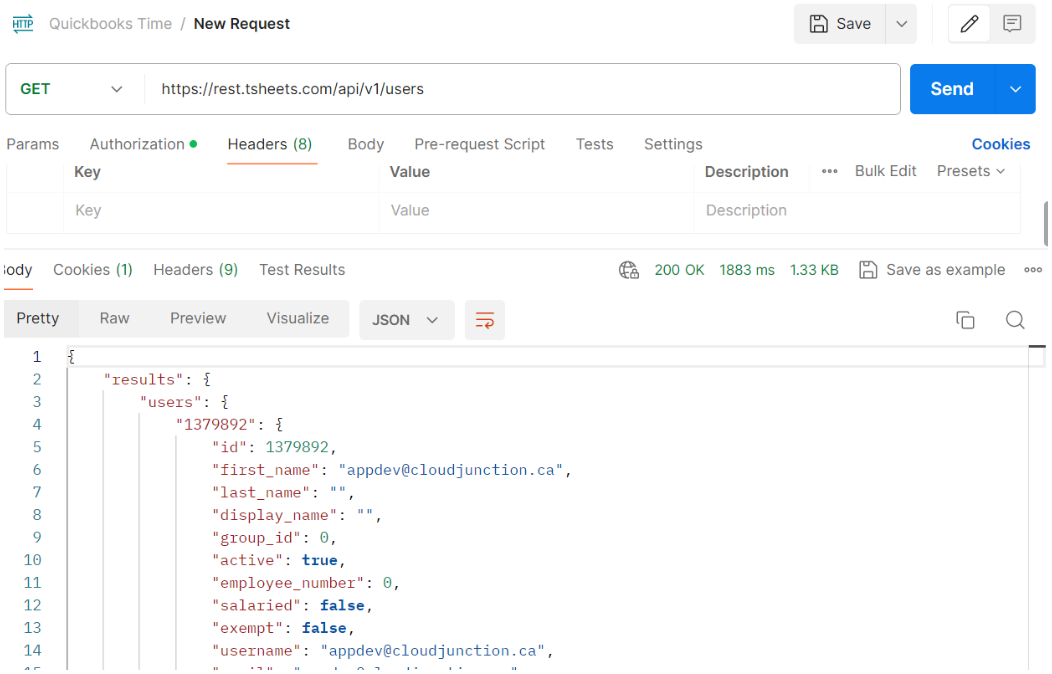
Introduction:

Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.

Use of Postman:

Almost any functionality that could be needed by any developer is encapsulated in this tool. It is used by over 5 million developers every month to make their API development easy and simple. It has the ability to make various types of HTTP requests(GET, POST, PUT, PATCH), save environments for later use, converting the API to code for various languages(like JavaScript, and Python).
Apart from this, If you want to take out Mocking API and user-friendly API documentation, this article offers an ideal option for you. Apidog is a powerful API development tool that connects everyone involved in the API development process. All-in-One workspace for API Design, Documentation, debugging, mocking, test.

Get Request:



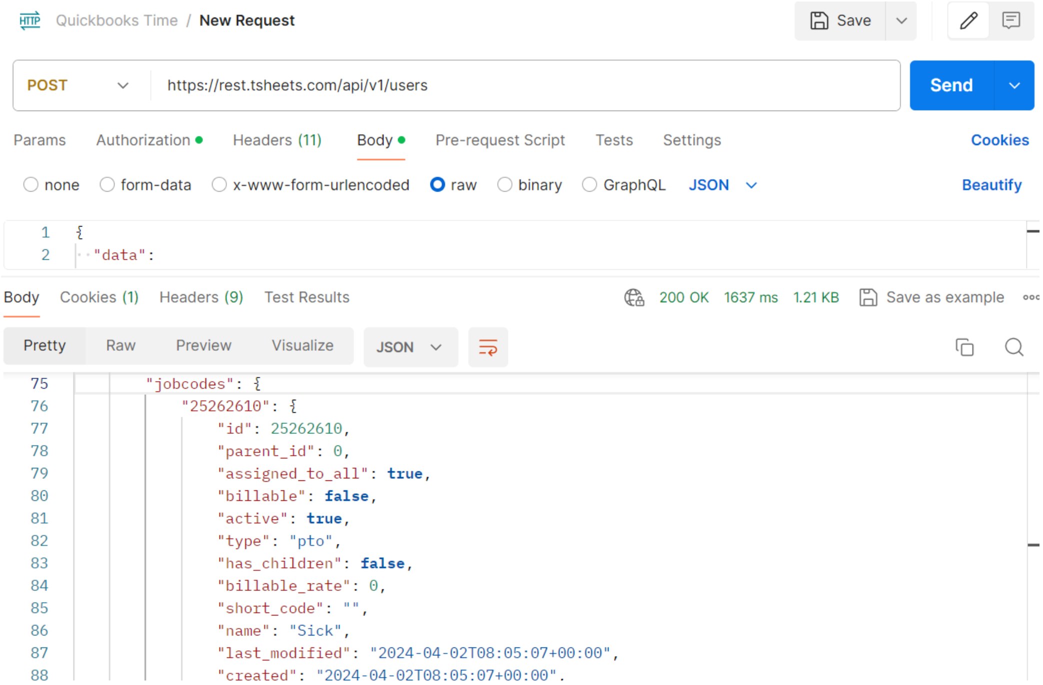
GET https://rest.tsheets.com/api/v1/users

Send

200 OK 1883 ms 1.33 KB

```
{
    "results": {
        "users": {
            "1379892": {
                "id": 1379892,
                "first_name": "appdev@cloudjunction.ca",
                "last_name": "",
                "display_name": "",
                "group_id": 0,
                "active": true,
                "employee_number": 0,
                "salaried": false,
                "exempt": false,
                "username": "appdev@cloudjunction.ca",
```

Post Request:



POST https://rest.tsheets.com/api/v1/users

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
  "data":
```

200 OK 1637 ms 1.21 KB

JSON

```
        "jobcodes": {
            "25262610": {
                "id": 25262610,
                "parent_id": 0,
                "assigned_to_all": true,
                "billable": false,
                "active": true,
                "type": "pto",
                "has_children": false,
                "billable_rate": 0,
                "short_code": "",
                "name": "Sick",
                "last_modified": "2024-04-02T08:05:07+00:00",
                "created": "2024-04-02T08:05:07+00:00",
```

We can also use put, patch, delete and other options but the software I used for accessing records through API does not give the option or allow to use these operations on their records.