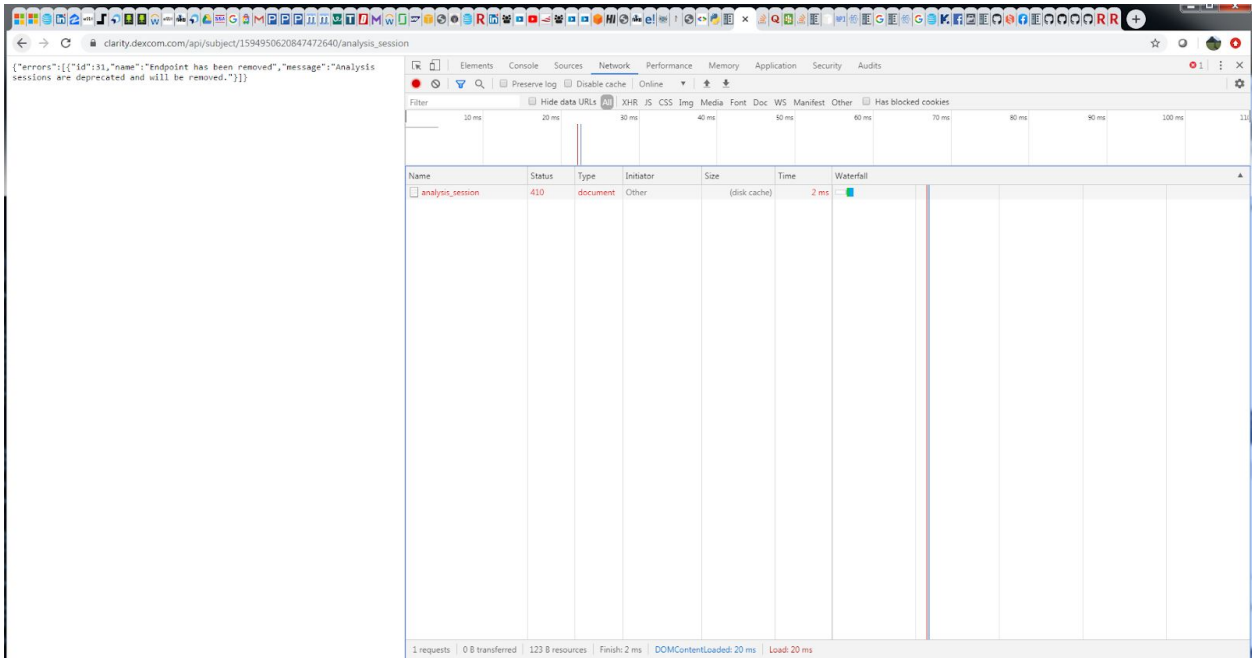
1. Reaching Endpoint in Browser

Executing manual steps to reach the endpoint in Chrome Browser, we see that the error message indicates that the document cannot be reached, because the ‘Endpoint has been removed’:

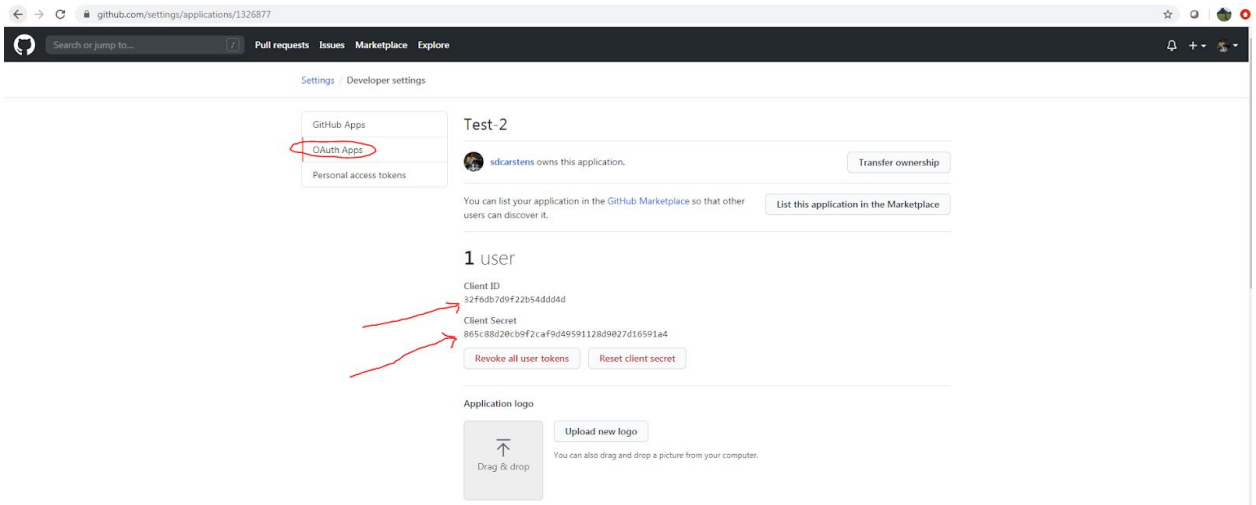


2. Managing OAuth2 session.

In order to establish the OAuth2 session, we need to have the ‘client_id’ and ‘client_secret’ of the application. Since these are not available for the Dexcom application, the example below establishes an OAuth2 session with my GitHub application:

```
########################################################################
#   OAuth2 login example with GitHub application
#
#   Based on workflow shown here:
https://requests-oauthlib.readthedocs.io/en/latest/examples/github.html
########################################################################
import time
from requests_oauthlib import OAuth2Session
```

```
# Credentials you get from registering a new GitHub application
client_id = '32f6db7d9f22b54ddd4d'
client_secret = '865c88d20cb9f2caf9d49591128d9027d16591a4'
print('clinet_id::::::::: ',client_id)
print('client_secret::::: ',client_secret)
print(' ')
```

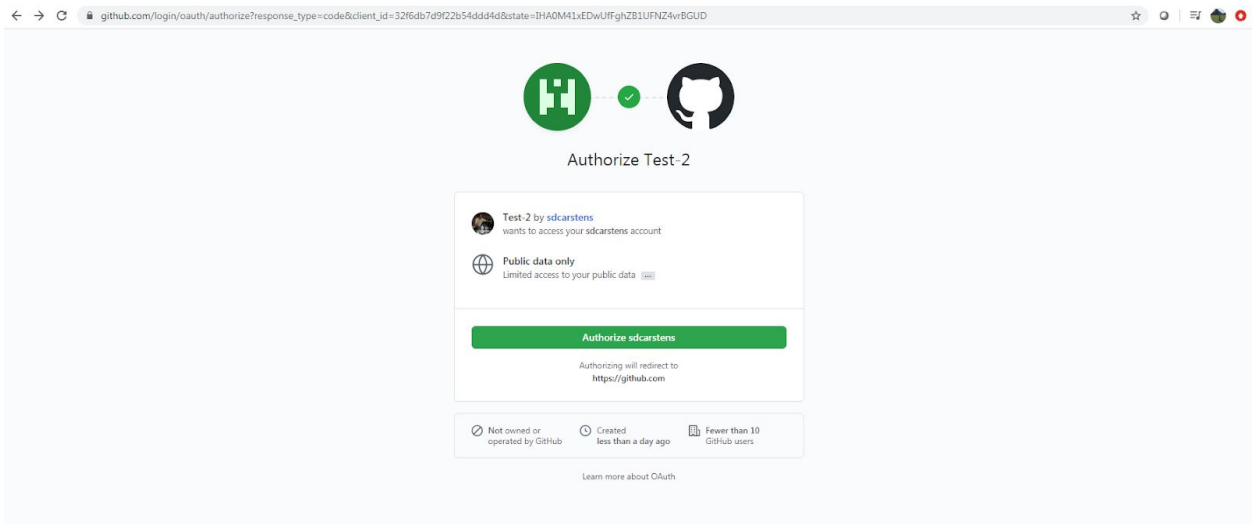



```
# OAuth endpoints given in the GitHub API documentation
authorization_base_url = 'https://github.com/login/oauth/authorize'
token_url              = 'https://github.com/login/oauth/access_token'

# Establish OAuth session to GitHub Application
github = OAuth2Session(client_id)
print('github:::::::: ',github)
print(' ')
```

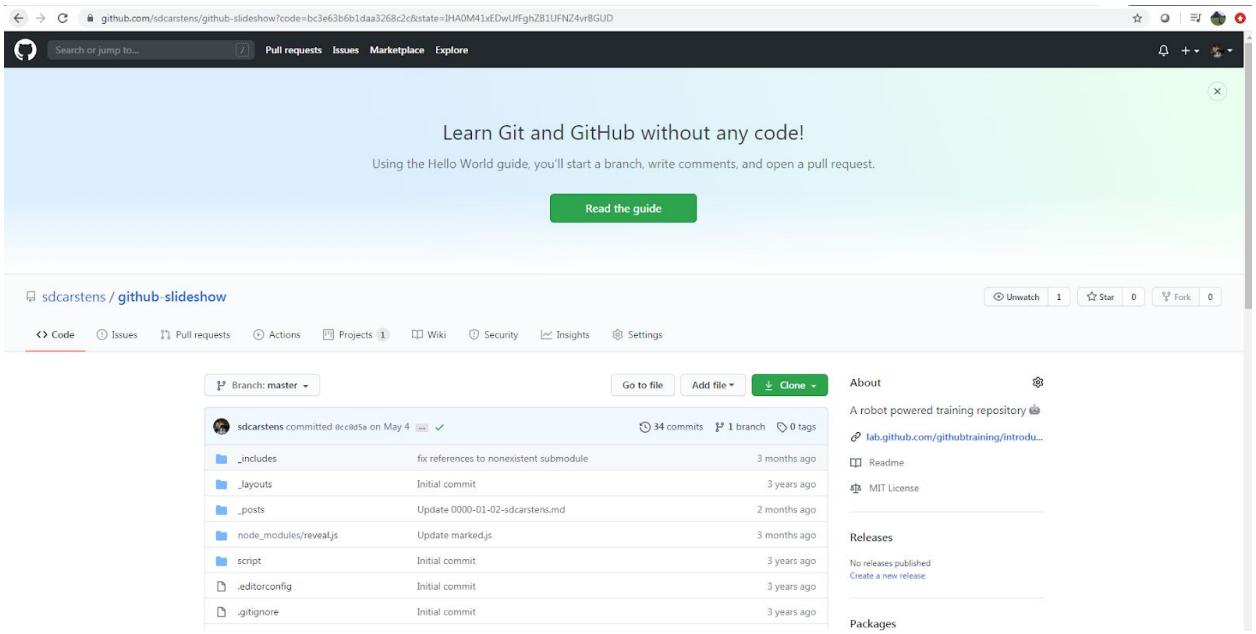
Redirect user to GitHub for Authorization

```
authorization_url, state = github.authorization_url(authorization_base_url)
print( 'Please go here and authorize,', authorization_url)
print(' ')
```



Get the authorization verifier code from the callback url

```
redirect_response = input('Paste the full redirect URL here:')
```

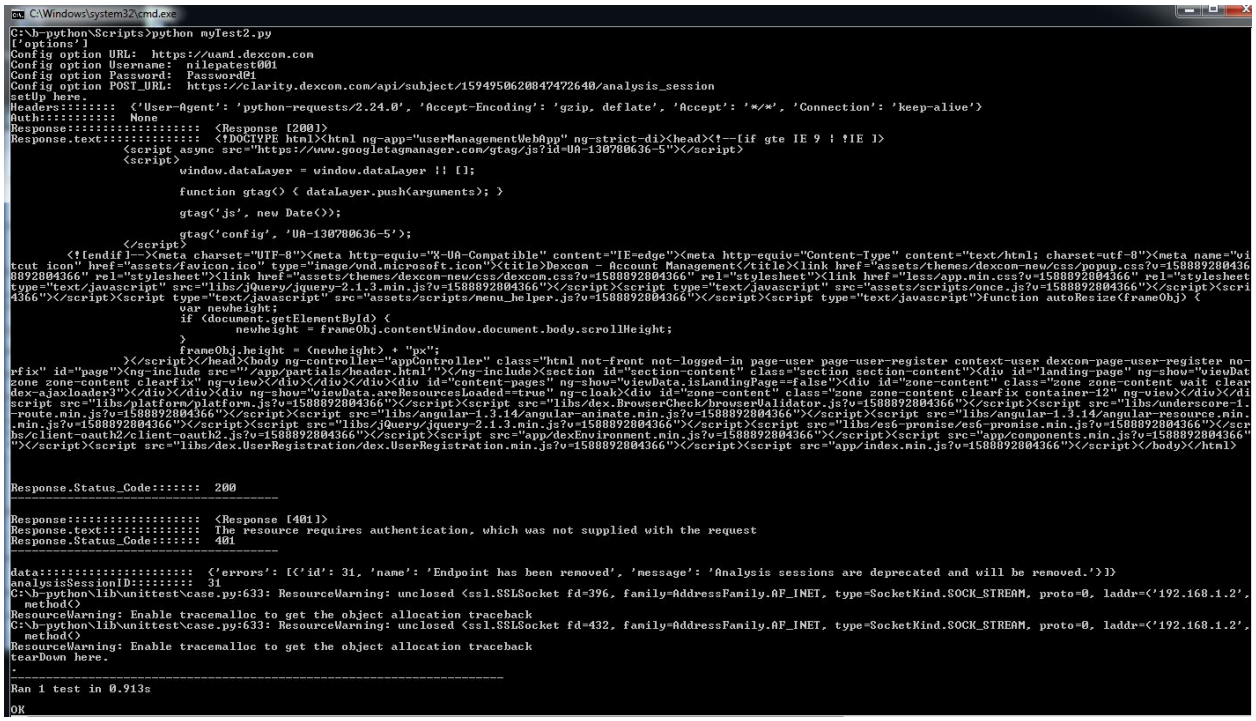


```
# Fetch the access token
github.fetch_token(token_url, client_secret=client_secret,
authorization_response=redirect_response)

# Fetch a protected resource, i.e. user profile
r = github.get('https://api.github.com/user')
print('Protected Resource Content:::::: ', r.content)
```

3. Test For Interacting with API HTTP Request calls

Below is the output from a Test which creates a Session object and executes POST and GET requests to the Dexcom REST API. The error message returned again says that the 'Endpoint has been removed'.



The Python code can be found here: https://github.com/sdcarstens/Dexcom-Test2

```
#############################################################################
#
#    Test 2 - API Automation with 'reqests' library and Session
#
#############################################################################
import time
import unittest
import configparser
from   selenium import webdriver
from   getpass import getpass

import locators
import element
import page

import requests
import json
from   requests.auth import HTTPBasicAuth
```

```
#
# Read in config options for URL, Username, Password
#
config = configparser.RawConfigParser()
config.read('myTest1.conf')

config_url       = config.get('options', 'url')
config_username  = config.get('options', 'username')
config_password  = config.get('options', 'password')
config_post_url  = config.get('options', 'post_url')

print(config.sections())
print("Config option URL: ",config_url)
print("Config option Username: ",config_username)
print("Config option Password: ",config_password)
print("Config option POST_URL: ",config_post_url)









class DexcomApiAutomation(unittest.TestCase):
    """Use the Sessions object class and unittest framework to test Dexcom API endpoint."""

    def setUp(self):
        print("setUp here.")
        #self.driver = webdriver.Chrome()
        #self.driver.get(config_url)

    def test_search_in_python_org(self):
        """
        Tests the ability to POST to
https://clarity.dexcom.com/api/subject/1594950620847472640/analysis_session
        """

        # Create new session object
        newSession = requests.Session()

        #
        #  Login to Dexcom URL
        #
        #  Expect status code 200, successful login
        #
        data = {"login":config_username, "password":config_password}
        auth = HTTPBasicAuth(config_username, config_password)
        r = newSession.post(url=config_url, data=data, auth=auth)


        print("Headers:::::::: ", newSession.headers)
        print("Auth:::::::::: ", newSession.auth)
        print("Response:::::::::::::::::: ",r)
```

```
        print("Response.text:::::::::::::: ",r.text)
        print("Response.Status_Code::::::: ",r.status_code)
        assert r.status_code == 200,  "Expected status code should be 200"
        print("----------------------------------------")
        print(" ")



        #
        #   Now use the POST request to test API endpoint
        #
        #   Expect response status_code 401 for invalid authentication
        #
        data = {'sender': 'Alice', 'receiver': 'Bob', 'message': 'We did it!'}
        headers = {'Content-type': 'application/json', 'Accept': 'text/plain'}
        r = newSession.post(config_post_url, data=json.dumps(data), headers=headers,
auth=auth)
        print("Response:::::::::::::::::: ",r)
        print("Response.text:::::::::::::: ",r.text)
        print("Response.Status_Code::::::: ",r.status_code)
        assert r.status_code == 401,  "Expected status code should be 401"
        print("----------------------------------------")
        print(" ")


        #
        #  Now use the GET request to test API endpoint
        #
        #  Expect response: {'errors': [{'id': 31, 'name': 'Endpoint has been removed',
'message': 'Analysis sessions are deprecated and will be removed.'}]}
        #
        r = newSession.get(url = config_post_url, auth=auth)
        data = r.json()
        print("data:::::::::::::::::::::: ",data)
        analysisSessionId = data['errors'][0]['id']
        print("analysisSessionID::::::::: ",analysisSessionId)
        assert analysisSessionId != 'None', "The value of analysisSessionId should not be
None"



    def tearDown(self):
        print("tearDown here.")
        #self.driver.close()

if __name__ == "__main__":
    unittest.main()
```