Setting Up and Running the FitPeo Revenue Calculator Script

This document provides step-by-step instructions to set up and run the FitPeoTestPage Selenium script for testing the revenue calculator on the FitPeo website.

Prerequisites

Python: Ensure Python (version 3.6 or higher) is installed on your machine. You can download Python from Hyperlink removed for security reasons.

Google Chrome: Ensure Google Chrome is installed, as the script uses Chrome WebDriver.

Step-by-Step Guide

1. Install Dependencies

You need to install the Selenium library for Python. You can install it using pip:

*pip install selenium*

2. Download the Script

Save the following script to a file named “FitPeoTestPage.py”.

*import time  
from selenium import webdriver  
from selenium.webdriver import ActionChains, Keys  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
  
  
class FitPeoTestPage:  
 def \_\_init\_\_(self, driver\_path='chromedriver'):  
 # Initialize the Chrome WebDriver and WebDriverWait  
 self.driver = webdriver.Chrome()  
 self.wait = WebDriverWait(self.driver, 10)  
  
 def open\_homepage(self, url):  
 # Open the given URL and adjust initial window size and position  
 self.driver.get(url)  
 self.driver.set\_window\_size(1552, 832)  
 self.driver.set\_window\_position(-2, 0)  
 time.sleep(1)  
  
 def navigate\_to\_revenue\_calculator(self):  
 # Navigate to the Revenue Calculator page by clicking the appropriate link  
 try:  
 revenue\_calculator\_link = self.wait.until(EC.element\_to\_be\_clickable((By.LINK\_TEXT, "Revenue Calculator")))  
 revenue\_calculator\_link.click()  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error navigating to Revenue Calculator: {e}")  
  
 def adjust\_window(self, width, height):  
 # Adjust the browser window size and position  
 try:  
 self.driver.set\_window\_size(width, height)  
 self.driver.set\_window\_position(-2, 0)  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error adjusting window: {e}")  
  
 def scroll\_to\_slider(self):  
 # Scroll the page to the slider section using JavaScript  
 try:  
 slider\_section = self.wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id=":r0:"]')))  
 self.driver.execute\_script("arguments[0].scrollIntoView({ behavior: 'smooth', block: 'center' });",  
 slider\_section)  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error scrolling to slider: {e}")  
  
 def set\_slider\_value(self, value\_set\_to):  
 # Set the slider to the desired value by calculating the pixel offset  
 try:  
 slider\_element = self.wait.until(EC.presence\_of\_element\_located(  
 (By.XPATH,  
 "(//span[@class='MuiSlider-root MuiSlider-colorPrimary MuiSlider-sizeMedium css-16i48op'])[1]")))  
 slider\_size = slider\_element.size  
 slider\_width = slider\_size['width']  
 slider\_value\_max = 2000  
  
 # Get the current value of the slider  
 slider\_current\_value\_element = self.wait.until(  
 EC.presence\_of\_element\_located(  
 (By.XPATH, "//input[@type='number' and contains(@class, 'MuiInputBase-input')]")))  
 current\_slider\_value = int(slider\_current\_value\_element.get\_attribute("value"))  
  
 # Calculate the required pixel movement based on the desired value  
 slider\_set\_value = value\_set\_to - current\_slider\_value  
 pixel\_move = (slider\_width / slider\_value\_max) \* slider\_set\_value  
  
 # Move the slider's thumb using the calculated pixel offset  
 slider\_thumb = self.wait.until(EC.element\_to\_be\_clickable((By.XPATH,  
 "//span[contains(@class,'MuiSlider-thumb MuiSlider-thumbSizeMedium MuiSlider-thumbColorPrimary css-1sfugkh')]")))  
 ActionChains(self.driver).drag\_and\_drop\_by\_offset(slider\_thumb, pixel\_move, 0).perform()  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error setting slider value: {e}")  
  
 def input\_number(self, number\_value):  
 # Input a number into the specified field after clearing its existing value  
 try:  
 input\_element = self.wait.until(EC.presence\_of\_element\_located(  
 (By.XPATH, "//input[@type='number' and contains(@class, 'MuiInputBase-input')]")))  
 input\_element.send\_keys(Keys.BACK\_SPACE \* 4)  
 input\_element.send\_keys(str(number\_value))  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error inputting number: {e}")  
  
 def click\_checkbox(self, cpt\_code, value):  
 # Click the checkbox corresponding to the specified CPT code and value  
 try:  
 checkbox = self.wait.until(EC.presence\_of\_element\_located(  
 (By.XPATH,  
 f"//p[text()='{cpt\_code}']/following::span[text()='{value}']/preceding::input[@type='checkbox'][1]")))  
 self.driver.execute\_script("arguments[0].scrollIntoView({ behavior: 'smooth', block: 'center' });",  
 checkbox)  
 checkbox.click()  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error clicking checkbox for {cpt\_code}: {e}")  
  
 def validate\_total\_reimbursement(self):  
 # Validate and print the total recurring reimbursement displayed on the page  
 try:  
 recuring\_reimbursement = self.wait.until(EC.presence\_of\_element\_located((By.XPATH,  
 "//div[@class='MuiBox-root css-m1khva']//p[@class='MuiTypography-root MuiTypography-body1 inter css-12bch19']")))  
 print(f"Total Recurring Reimbursement: {recuring\_reimbursement.text}")  
 time.sleep(1)  
  
 # Validate and print the reimbursement value from the header  
 header\_element = self.wait.until(  
 EC.presence\_of\_element\_located((By.XPATH, "//p[position()=4]//p[position()=1]")))  
 print(  
 f"Header displaying 'Total Recurring Reimbursement for all Patients Per Month:' shows the value: {header\_element.text}, and the expected value is $110700")  
 time.sleep(1)  
 except Exception as e:  
 print(f"Error validating total reimbursement: {e}")  
  
 def close\_browser(self):  
 # Close the browser  
 self.driver.quit()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Create an instance of the FitPeoTestPage class and execute the test steps  
 test = FitPeoTestPage()  
  
 # Steps of the test  
 test.open\_homepage("https://www.fitpeo.com/")  
 test.navigate\_to\_revenue\_calculator()  
 test.adjust\_window(412, 915)  
 test.scroll\_to\_slider()  
 #time.sleep(2)  
 test.set\_slider\_value(820)  
 #time.sleep(2)  
 test.adjust\_window(1552, 832)  
 test.input\_number(560)  
 #time.sleep(2)  
 test.click\_checkbox('CPT-99091', '57')  
 test.click\_checkbox('CPT-99453', '19.19')  
 test.click\_checkbox('CPT-99454', '63')  
 test.click\_checkbox('CPT-99474', '15')  
 test.validate\_total\_reimbursement()  
 test.close\_browser()*

3. Run the Script

To run the script, execute the following command in your terminal/command prompt/:

*python fitpeo\_revenue\_calculator.py*

Or you can open the saved file in python IDE/Pycharm

**How the Script Works**

**Initialization:**

The script initializes the Chrome WebDriver and sets up WebDriverWait for explicit waits.

**Open Homepage:**

The script opens the FitPeo homepage and sets the initial window size and position.

**Navigate to Revenue Calculator:**

It clicks on the "Revenue Calculator" link to navigate to the calculator page.

**Adjust Window:**

The script adjusts the browser window size to specified dimensions.

**Scroll to Slider:**

It scrolls down to the slider element on the page.

**Set Slider Value:**

The script sets the slider to the desired value by calculating the necessary pixel movement.

**Input Number:**

It inputs a number value into the provided number field after clearing the existing content.

**Click Checkboxes:**

The script clicks the checkboxes corresponding to specified CPT codes and values.

**Validate Total Reimbursement:**

It validates and prints the total recurring reimbursement displayed on the page.

**Close Browser:**

Finally, the script closes the browser.

**Note**

In the script, a delay *(time.sleep(2))* is added after each action to allow you to observe the actions being performed. To reduce the script's execution time, you can comment out all instances of *time.sleep(2)* in the script.

**Troubleshooting**

**Errors**: If you encounter any errors, check the console output for detailed error messages. Ensure all XPaths and element locators are correct.

Software and Versions Used

Python: Version 3.11.9

Google Chrome: Version 116 or later

Selenium: Version 4.0 or later

Conclusion

Following these steps will help you set up and run the FitPeo revenue calculator script using Selenium WebDriver. Modify the script as necessary to accommodate changes in the web page structure or different test scenarios.