دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

آزمون واسط کاربری

UI Testing

پدیدآورنده:

محمد امین کیانی

4003613052

دانشجوی کارشناسی، دانشکده‌ی کامپیوتر، دانشگاه اصفهان، اصفهان،
Aminkianiworkeng@gmail.com

استاد درس: جناب اقای دکتر شعرباف

نیمسال اول تحصیلی 04-1403

فهرست مطالب

مستندات

لینک ویدیو

https://drive.google.com/file/d/1UCS4HVLe9EOMyuH27Vo2tuQaaMe6cURE/view?usp=sharing

بررسی خطاهای متداول کلی

1- در بعضی از موارد بعضی از سایت ها، تست ها ارور مربوط به کش دارند(مثال هنگامی که سایت لود شد و صفحه privacy&Cookies میاد که باید ریجت یا اکسپت کرد و سپس ادامه تست را اجرا کرد.):

The page keeping the extension port is moved into back/forward cache, so the message channel is closed.

چون به اطلاعات آن دکمه نمیتوان دسترسی پیدا کرد و تنها بار اول که سایت را باز میکنیم، پیدا می‌شود پس دیگر نخواهد بود مگر با پاک کردن کوکی ها شدنی‌است.

2- در بعضی از موارد همه ی سایت ها، تست ها ارور مربوط به منقضی شدن جستوجو دارند(مثال هنگامی که سایت لود شد و دکمه ی مورد نظر را می خواهد پیدا کند اما نمی‌تواند.):

Trying to find css=.order-number > strong... Failed:16:55:41 Implicit Wait timed out after 30000ms

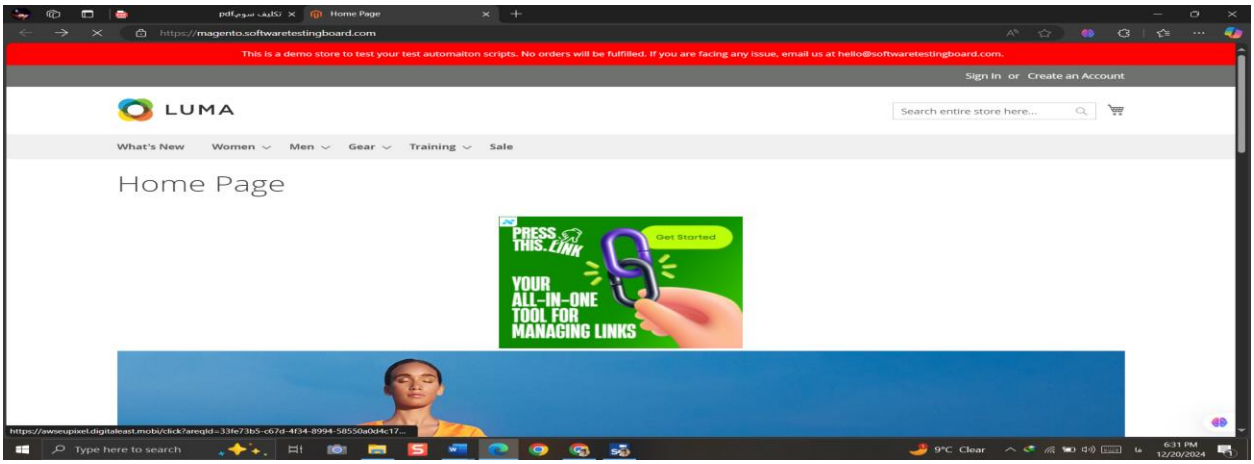
- **'Buy a new stuff testing' ended with 1 error(s)**

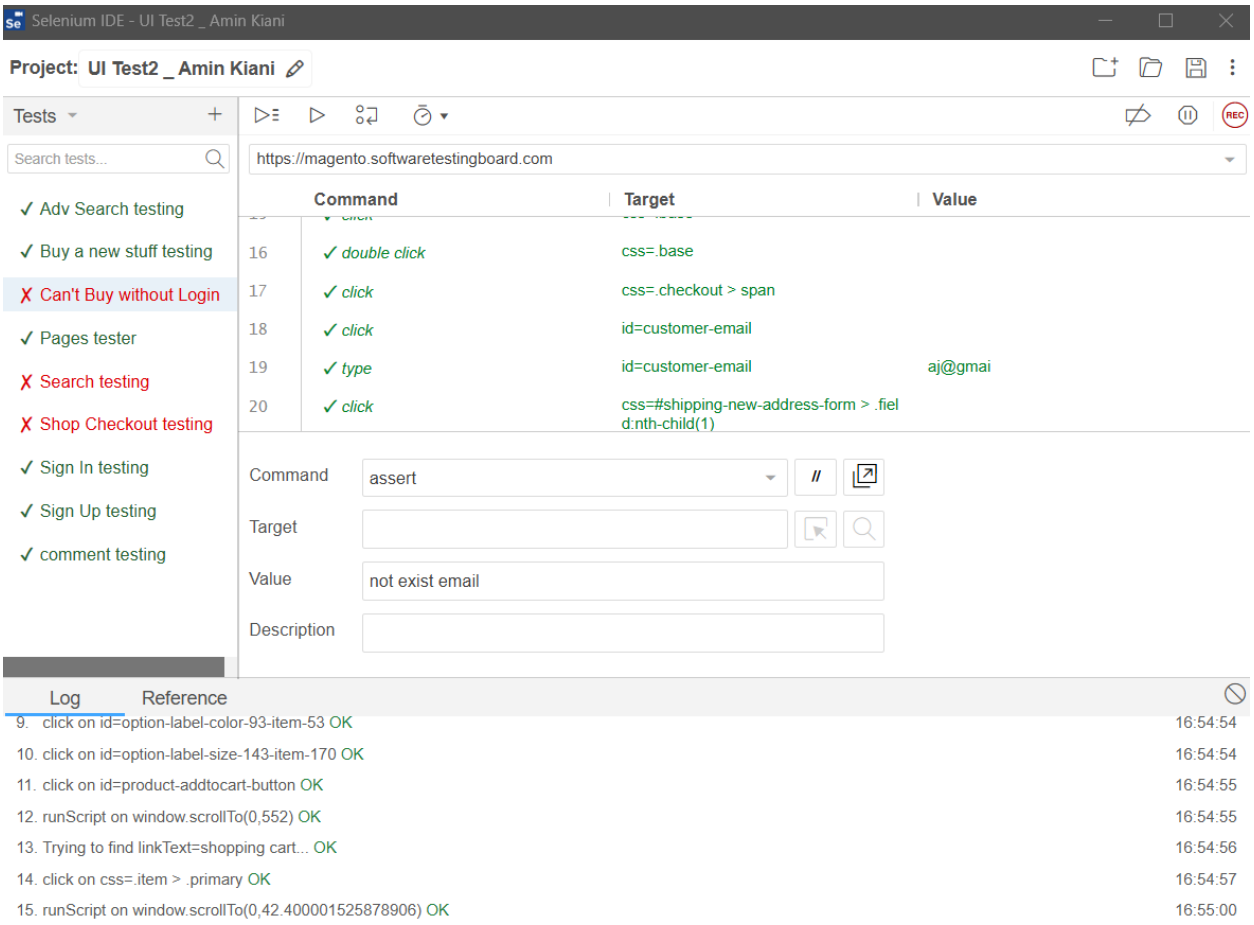
گاهی با تغییر هدف در کامند مربوطه می توان جهت گیری دکمه یابی را تغییر داده و دسترسی به آن پیدا کرد و حتی می‌توان در برخی موارد از دابل کلیک فیدبک درستی از موقعیت مکانی آن ایجاد کرد اما در حالت کلی مشکل پیدا شدن دکمه ها به تعداد دفعات زیادی وجود دارد.

تکلیف سوم

بخش اول:

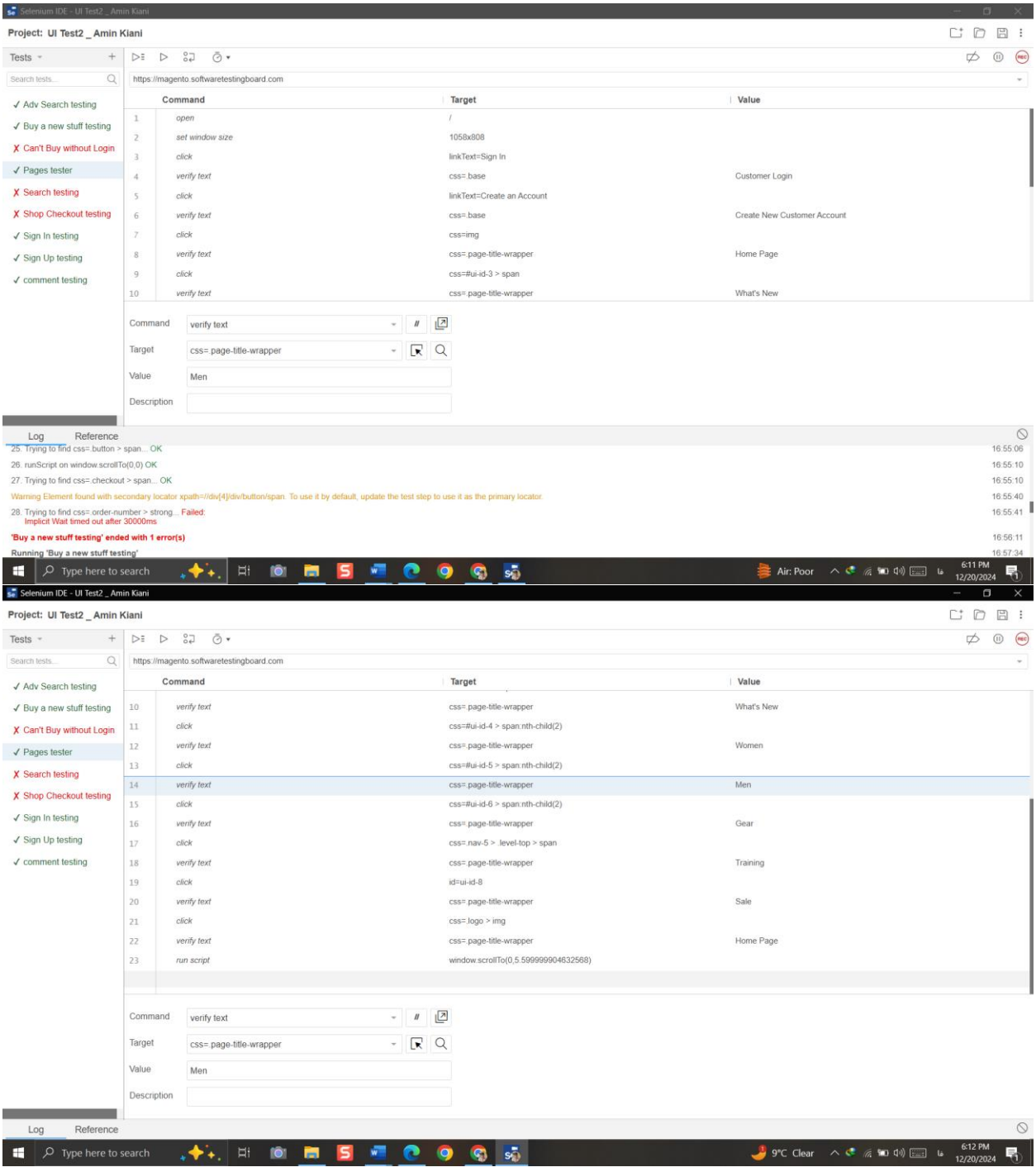
در ابتدا سایت شماره7 را انتخاب کرده و 9 تست مهم از عملکرد کلی آن رکورد کردم که 5 مورد را در ادامه شرح می‌دهم:





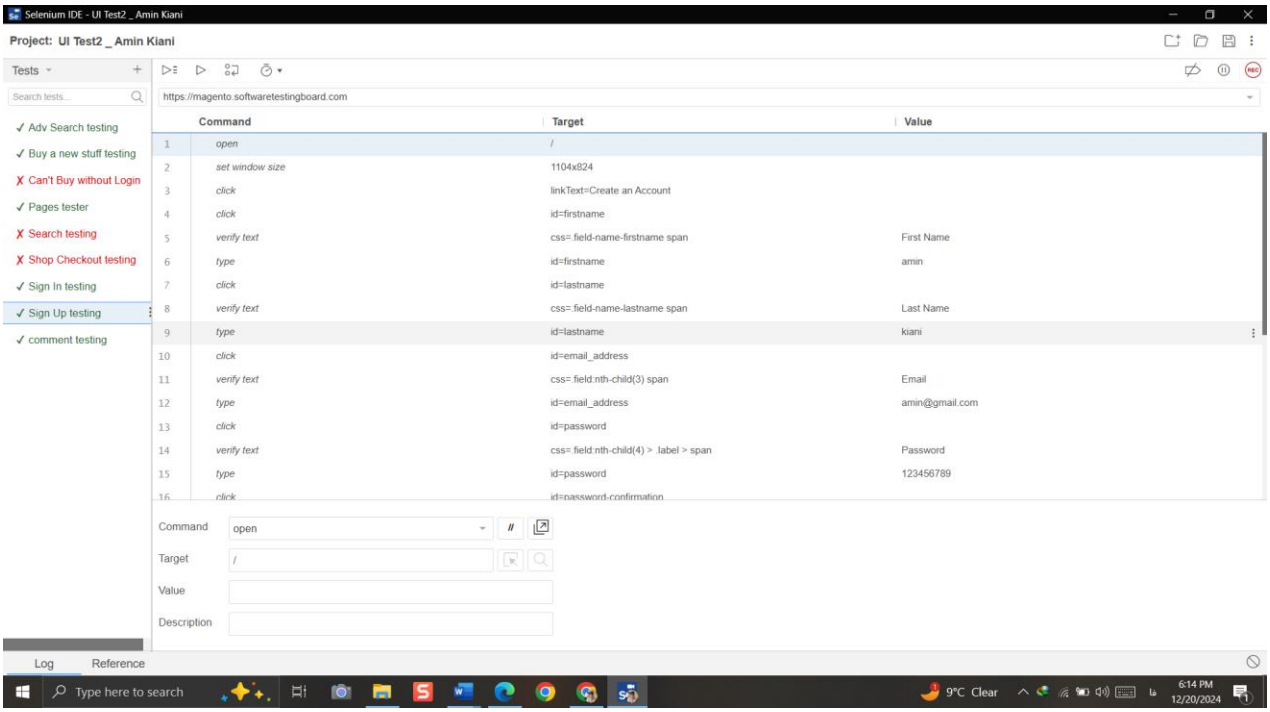
سناریو 1: بررسی کلی صفحات هدر

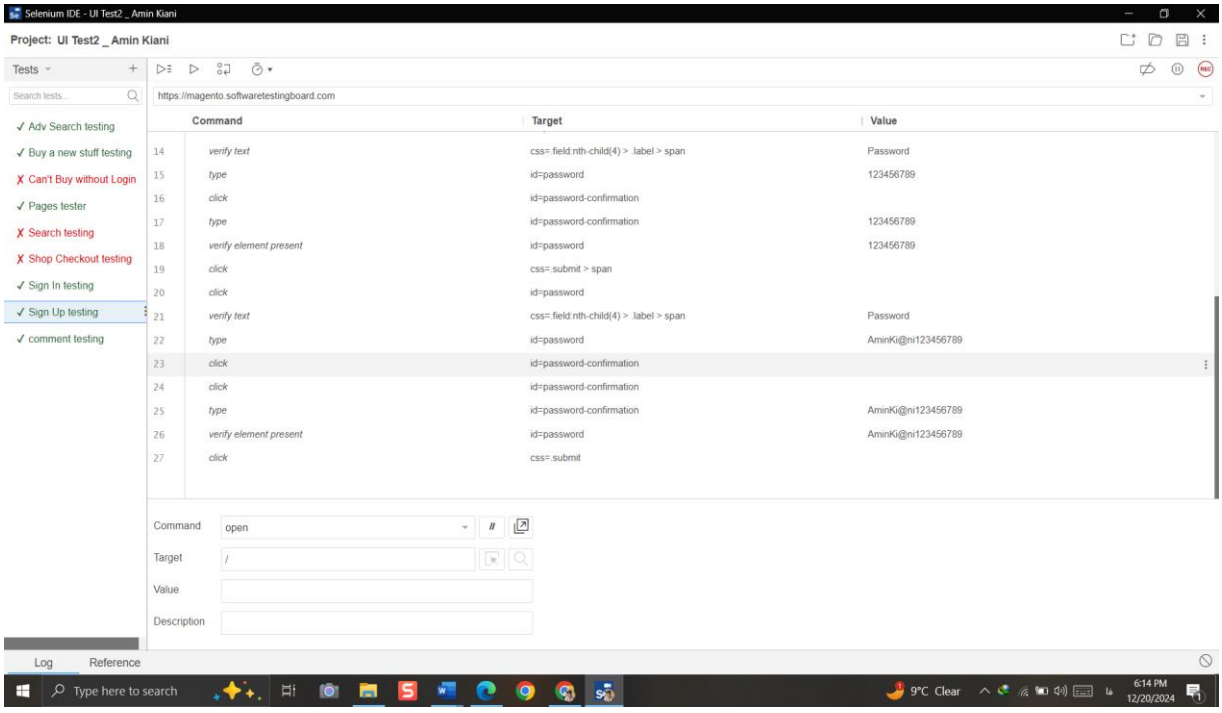
عملکرد دکمه‌های هدر سایت و صفحات کلی آن تست شده تا صحت لینک بودن آن‌ها تست شود.



سناریو 2: ثبت‌نام در سایت

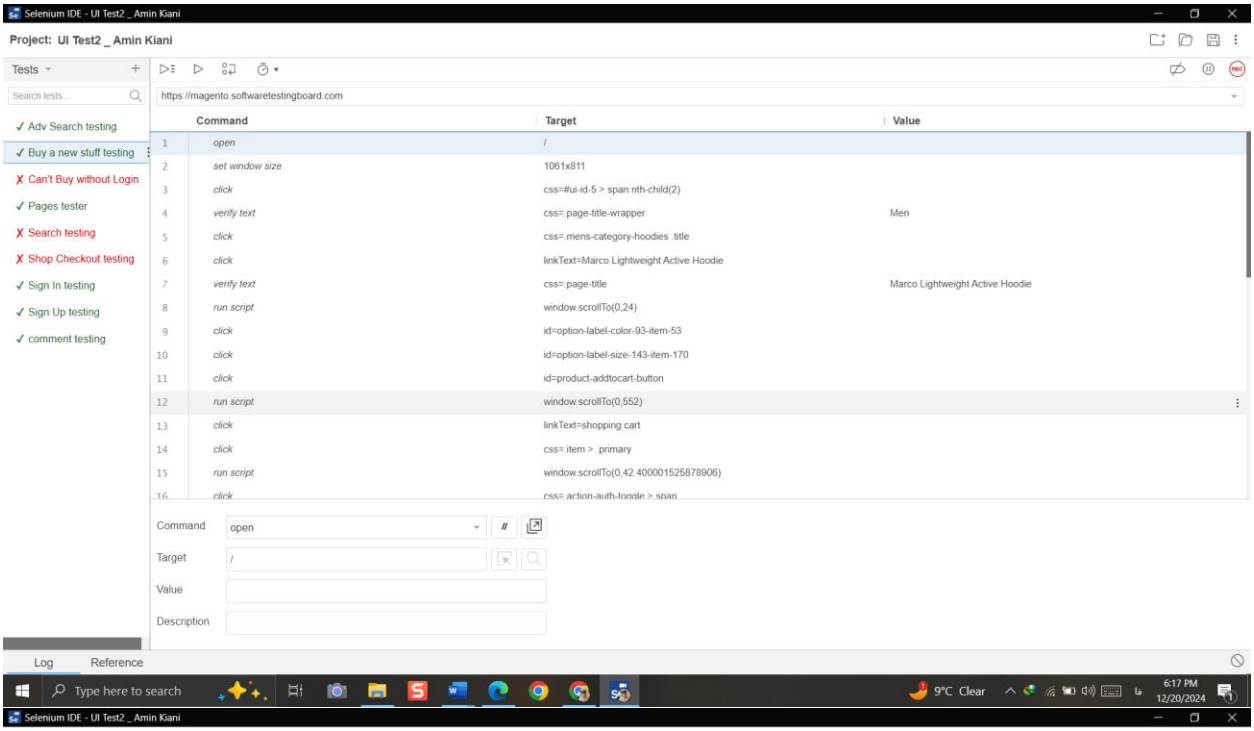
عملکرد دکمه‌های این بخش و صحت تایپ‌ها و اعتبارسنجی بررسی شده و تا یک کاربر ثبت شود و در صورت وجود این اطلاعات از قبل به درستی ارور تکراری بودن اطلاعات را با توجه به کوکی‌ها می‌دهد.

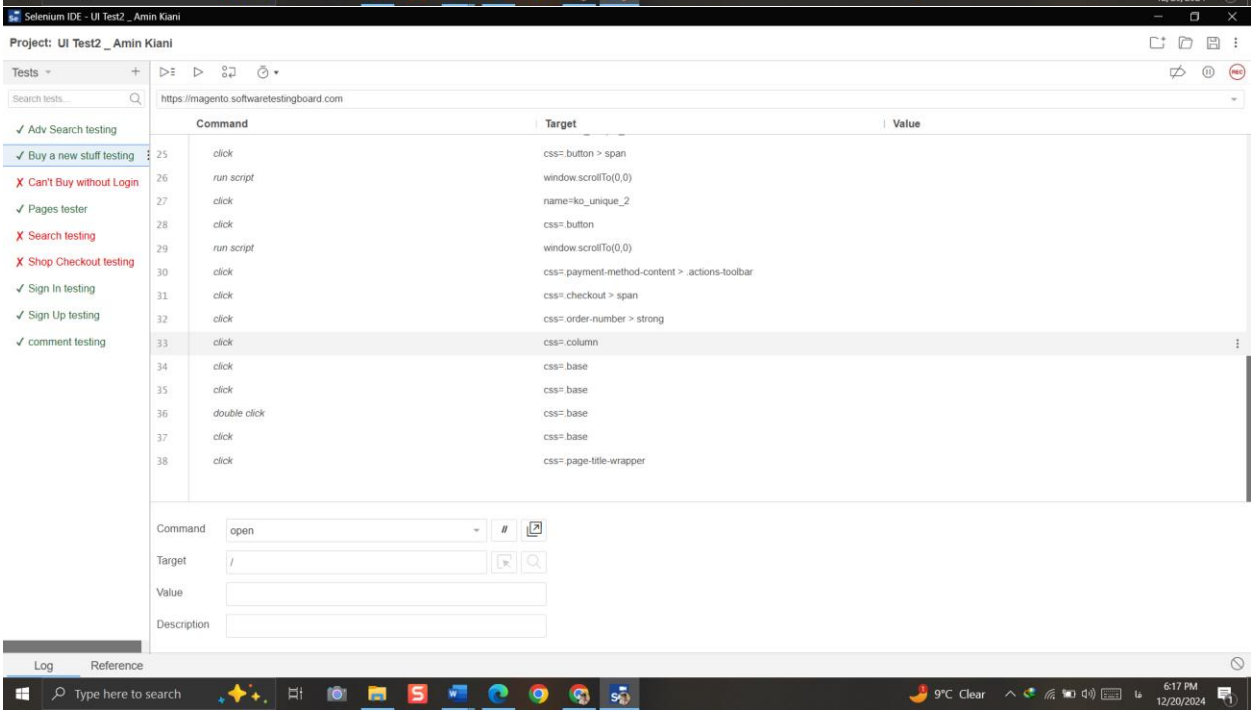




سناریو 3: خرید یک کالای جدید

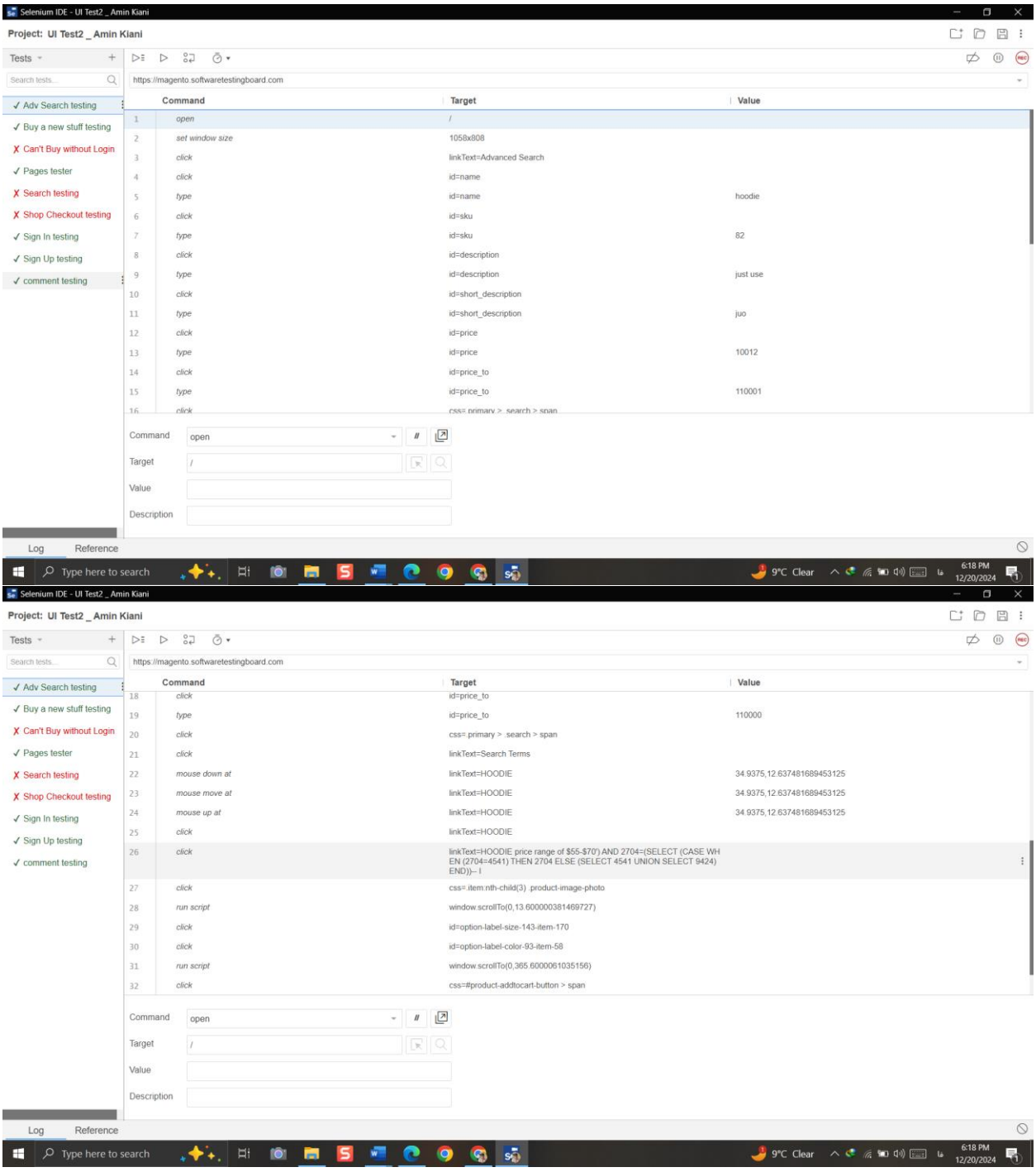
در این بخش صفر تا صد مراحل خرید یک کالا همراه با لاگین کردن برای توان ثبت کالا و دریافت کد معتبر سفارش انجام می‌شود که یک تست اضافی و کمکی نیز در فایل سلنیوم وارد کردم تا عدم توان خرید توسط ثبتنام نکرده‌ها نیز بررسی شود.





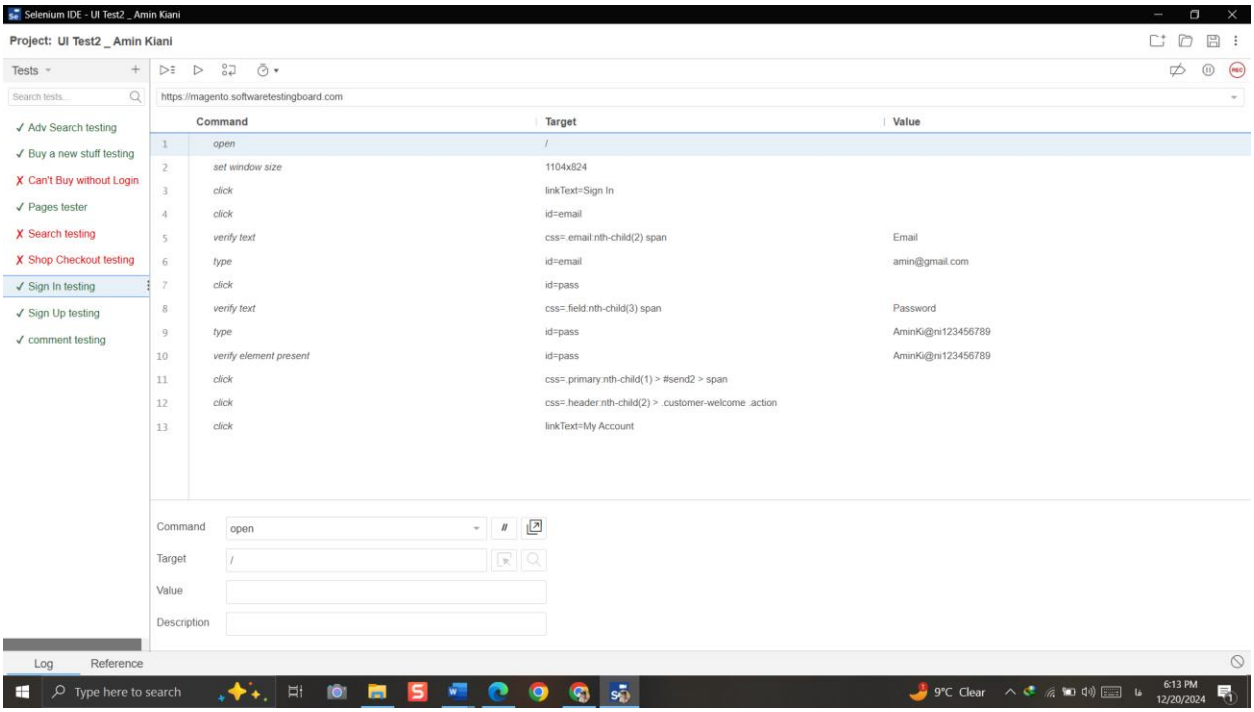
سناریو 4: جستوجوی کالا

در این بخش به بررسی جستوجوی پیشرفته برای یافتن کالای مورد نظر پرداخته شده که در کنار این بخش نیز یک تست سرچ معمولی توسط نوار منو انجام شده که هر دو به درستی انجام می‌شوند.



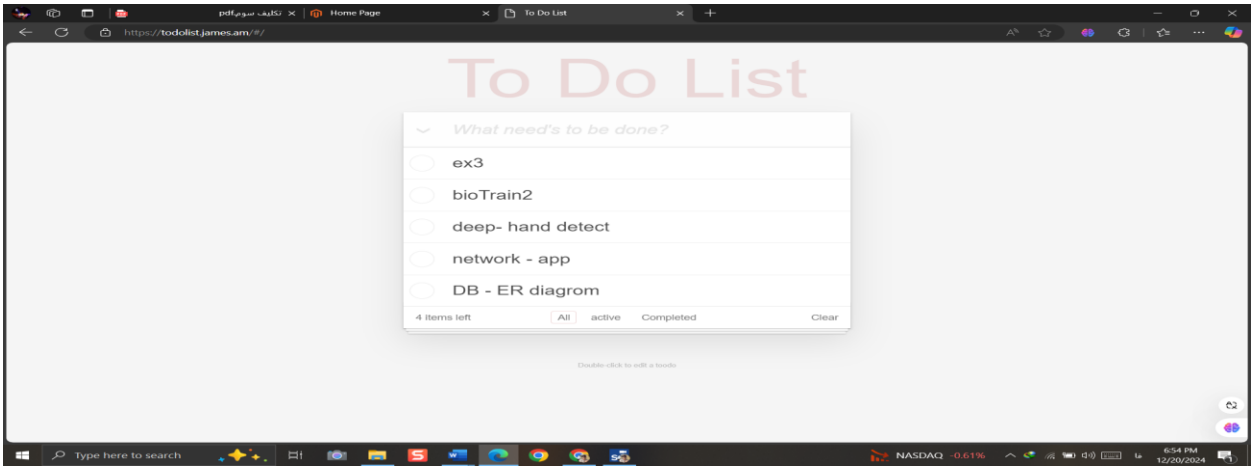
سناریو 5: ورود به اکانت شخصی

در این بخش نیز با وارد کردن اطلاعات سعی در لاگین شدن دارد که در صورت رجیستری بودن به درستی وارد شده و عدم وجود چنین اکانتی سبب جلوگیری از ورود و ارور تست می‌شود.



بخش دوم:

در ابتدا سایت شماره8 را انتخاب کرده و 5 تست مهم از عملکرد کلی آن رکورد کردم که در ادامه شرح می‌دهم:



ابزارها:

- **Unittest :**چارچوب تست استاندارد پایتون.
- **WebDriverWait و Selenium** : ابزار خودکارسازی مرورگر.
- **EC.presence_of_element_located** : برای اطمینان از این که عنصر در DOM بارگذاری شده است.
- **time.sleep** : برای ایجاد تأخیرهای زمانی و مشاهده‌ی تغییرات UI.

```
from selenium import webdriver
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
import unittest
import time


class TodoAppTests(unittest.TestCase):
    URL = "https://todolist.james.am/"

    def setUp(self):
        """Set up the browser before each test."""
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        self.driver.get(self.URL)
        self.wait = WebDriverWait(self.driver, 10)

    def tearDown(self):
        """Close the browser after each test."""
        self.driver.quit()

    def add_tasks(self, tasks):
        """Helper function to add multiple tasks."""
        input_box = self.wait.until(
            EC.presence_of_element_located((By.CSS_SELECTOR,
"input.new-todo"))
        )
        for task in tasks:
            input_box.send_keys(task + Keys.RETURN)
            time.sleep(1)  # Delay after adding each task

    def test_add_to_do(self):
        """Test adding tasks and verifying their count."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
```

```
        self.add_tasks(tasks)

        # Verify the number of tasks added
        time.sleep(2)  # Delay before verification
        list_items = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li")
        self.assertEqual(len(list_items), len(tasks),
                         f"Expected {len(tasks)} tasks, but found
{len(list_items)}")

    def test_check_items_left(self):
        """Test the remaining items count."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
        self.add_tasks(tasks)

        # Complete one task
        first_task_checkbox = driver.find_element(By.CSS_SELECTOR,
"ul.todo-list li:first-child .toggle")
        first_task_checkbox.click()
        time.sleep(2)  # Delay after clicking checkbox

        # Verify the remaining tasks count
        remaining_count = driver.find_element(By.CSS_SELECTOR,
"footer.footer .todo-count strong").text
        self.assertEqual(int(remaining_count), len(tasks) - 1,
                         f"Expected remaining count to be
{len(tasks) - 1}, but found {remaining_count}")

    def test_check_filter(self):
        """Test filtering tasks by 'active' and 'completed'."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
        self.add_tasks(tasks)

        # Complete some tasks
        driver.find_element(By.CSS_SELECTOR, "ul.todo-list
li:first-child .toggle").click()
        time.sleep(1)  # Delay after completing a task
        driver.find_element(By.CSS_SELECTOR, "ul.todo-list li:last-
child .toggle").click()
        time.sleep(1)

        # Check active filter
        driver.find_element(By.CSS_SELECTOR, 'ul.filters li
a[href="#/active"]').click()
        time.sleep(2)  # Delay after switching to active filter
        active_todos = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li")
```

```
        active_texts = [todo.text for todo in active_todos]
        self.assertEqual(active_texts, ["task 2"], "Active filter
did not return correct tasks.")

        # Check completed filter
        driver.find_element(By.CSS_SELECTOR, 'ul.filters li
a[href="#/completed"]').click()
        time.sleep(2)  # Delay after switching to completed filter
        completed_todos = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li")
        completed_texts = [todo.text for todo in completed_todos]
        self.assertEqual(set(completed_texts), {"task 1", "task
3"},
                         "Completed filter did not return correct
tasks.")

    def test_delete_item(self):
        """Test deleting a task."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
        self.add_tasks(tasks)

        # Delete the first task
        first_task = driver.find_element(By.CSS_SELECTOR, "ul.todo-
list li:first-child")
        delete_button = first_task.find_element(By.CSS_SELECTOR,
"button.destroy")
        driver.execute_script("arguments[0].click();",
delete_button)
        time.sleep(2)  # Delay after deleting task

        # Verify the task is deleted
        remaining_tasks = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li label")
        remaining_texts = [task.text for task in remaining_tasks]
        self.assertNotIn("task 1", remaining_texts, "Task 'task 1'
was not deleted.")

    def test_clear_completed(self):
        """Test clearing all completed tasks."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
        self.add_tasks(tasks)

        # Complete all tasks
        toggles = driver.find_elements(By.CSS_SELECTOR, "ul.todo-
list li .toggle")
        for toggle in toggles:
            toggle.click()
```

```
            time.sleep(1)  # Delay after completing each task

        # Clear completed tasks
        driver.find_element(By.CSS_SELECTOR, "button.clear-
completed").click()
        time.sleep(2)  # Delay after clearing tasks

        # Verify all tasks are cleared
        remaining_tasks = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li")
        self.assertEqual(len(remaining_tasks), 0, "Completed tasks
were not cleared.")

    def test_toggle_all(self):
        """Test toggling all tasks as completed."""
        driver = self.driver
        tasks = ["task 1", "task 2", "task 3"]
        self.add_tasks(tasks)

        # Use JavaScript to click the toggle-all checkbox to avoid
interception
        toggle_all = driver.find_element(By.CSS_SELECTOR,
"input#toggle-all")
        driver.execute_script("arguments[0].click();", toggle_all)

        # Wait a moment to allow UI updates
        import time
        time.sleep(2)

        # Verify all tasks are marked as completed
        completed_todos = driver.find_elements(By.CSS_SELECTOR,
"ul.todo-list li.completed")
        self.assertEqual(len(completed_todos), len(tasks),
                         "Not all tasks were marked as completed.")


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

کلاس بالا، شامل چند تست برای بررسی قابلیت‌های یک برنامه‌ی مدیریت وظایف (To-Do List) است.
که سناریوهای آن به شرح زیر اند:

test_add_to_do

- **هدف :**تست اضافه کردن وظایف به لیست.
- **کارکرد:**
  - چند وظیفه به لیست اضافه می‌شود.
  - بررسی می‌کند که تعداد وظایف موجود در لیست برابر با تعداد وظایف اضافه‌شده باشد.
- **موفقیت :**زمانی که تعداد وظایف موجود در DOM با تعداد وظایف اضافه‌شده برابر باشد.

test_check_items_left

- **هدف :**بررسی تعداد وظایف باقی‌مانده (وظایفی که هنوز تکمیل نشده‌اند).
- **کارکرد:**
  - چند وظیفه به لیست اضافه می‌شود.
  - یکی از وظایف به‌عنوان تکمیل‌شده علامت‌گذاری می‌شود.
  - تعداد وظایف باقی‌مانده بررسی می‌شود و با مقدار مورد انتظار مقایسه می‌شود.
- **موفقیت :**زمانی که شمارنده‌ی وظایف باقی‌مانده مقدار درست را نمایش دهد.

test_check_filter

- **هدف :**بررسی فیلتر کردن وظایف بر اساس وضعیت ("فعال" یا "تکمیل‌شده").
- **کارکرد:**
  - وظایفی اضافه می‌شوند.
  - تعدادی از وظایف به‌عنوان تکمیل‌شده علامت‌گذاری می‌شوند.
  - فیلتر "فعال" انتخاب می‌شود و بررسی می‌کند که فقط وظایف فعال نمایش داده شوند.
  - فیلتر "تکمیل‌شده" انتخاب می‌شود و بررسی می‌کند که فقط وظایف تکمیل‌شده نمایش داده شوند.
- **موفقیت :**زمانی که هر فیلتر، وظایف درست را نمایش دهد.

test_delete_item

- **هدف :**تست حذف یک وظیفه از لیست.
- **کارکرد:**
  - چند وظیفه به لیست اضافه می‌شود.
  - اولین وظیفه حذف می‌شود.
  - بررسی می‌کند که وظیفه‌ی حذف‌شده دیگر در لیست وجود نداشته باشد.
- **موفقیت :**زمانی که وظیفه‌ی حذف‌شده دیگر در DOM وجود نداشته باشد.

test_clear_completed

- **هدف :**تست پاک کردن تمام وظایف تکمیل‌شده.
- **کارکرد:**
  - چند وظیفه به لیست اضافه می‌شود.
  - تمام وظایف به‌عنوان تکمیل‌شده علامت‌گذاری می‌شوند.
  - با استفاده از دکمه‌ی "پاک کردن تکمیل‌شده‌ها"، تمام وظایف پاک می‌شوند.
  - بررسی می‌کند که لیست وظایف خالی باشد.
- **موفقیت :**زمانی که تمام وظایف تکمیل‌شده از DOM حذف شوند.

test_toggle_all

- **هدف :**تست علامت‌گذاری همه‌ی وظایف به‌عنوان تکمیل‌شده.
- **کارکرد:**
  - چند وظیفه به لیست اضافه می‌شود.
  - از دکمه‌ی "انتخاب همه" استفاده می‌شود تا همه‌ی وظایف به‌عنوان تکمیل‌شده علامت‌گذاری شوند.
  - بررسی می‌کند که تمام وظایف در DOM دارای کلاس completed.باشند.
- **موفقیت :**زمانی که همه‌ی وظایف به‌عنوان تکمیل‌شده در DOM نمایش داده شوند.