



American International University-Bangladesh (AIUB)

Department of Computer Science

Faculty of Science & Technology (FST)

Summer 22-23

Section: A

Software Quality and Testing

Selenium Software Testing

A Report submitted

By

SN	Student Name	Student ID
1	Ananya Chowdhury	18-39028-3
2	Tanzima Zahir	18-38958-3
3	MD. Ashraf Hossain Rifat	18-38928-3
4	Israt Jahan Pritha	18-39023-3

Under the supervision of

Abhijit Bhowmik

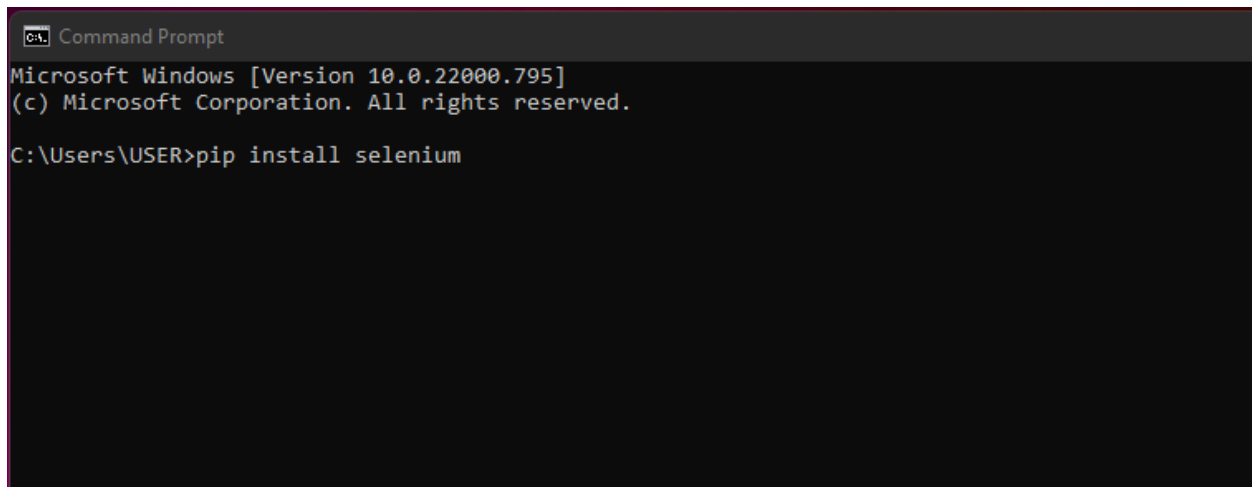
Department of Computer Science

Faculty of Science and Information Technology

American International University-Bangladesh

Software: Selenium

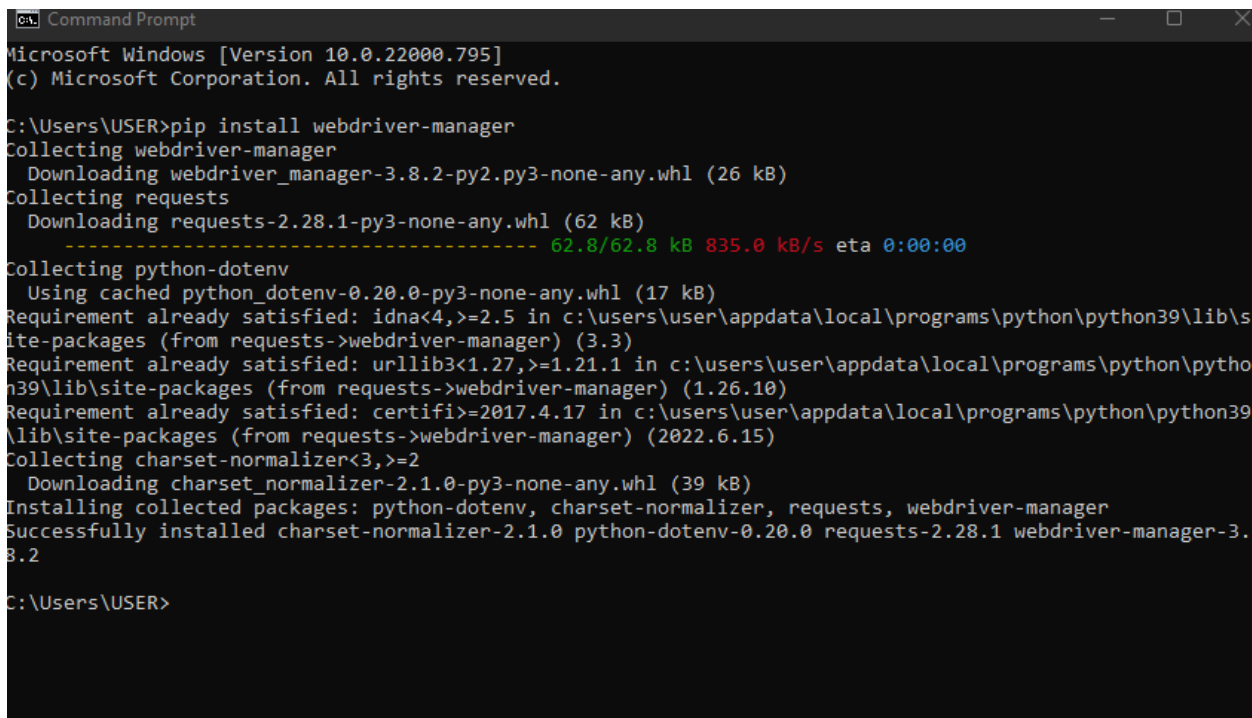
Installing Selenium: Before installing Selenium, we have to install Python first. As we have installed Python in our pc so we just skip that. Now for installing selenium we open Command Prompt and use this command - "pip install selenium"(shown in figure 1). After successfully installed selenium there would be appear a message like "Requirement already satisfied"(figure 2).



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>pip install selenium
```

Figure 1: Selenium installation in CMD



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.22000.795]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>pip install webdriver-manager
Collecting webdriver-manager
  Downloading webdriver_manager-3.8.2-py2.py3-none-any.whl (26 kB)
Collecting requests
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
----- 62.8/62.8 kB 835.0 kB/s eta 0:00:00
Collecting python-dotenv
  Using cached python_dotenv-0.20.0-py3-none-any.whl (17 kB)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\user\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (1.26.10)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\appdata\local\programs\python\python39\lib\site-packages (from requests->webdriver-manager) (2022.6.15)
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.0-py3-none-any.whl (39 kB)
Installing collected packages: python-dotenv, charset-normalizer, requests, webdriver-manager
Successfully installed charset-normalizer-2.1.0 python-dotenv-0.20.0 requests-2.28.1 webdriver-manager-3.8.2

C:\Users\USER>
```

Figure 2: Webdriver manager installation

Now we have to download Google Chrome driver as per our Google Chrome version. We have download Chromedriver_win32. After that we paste the Chromedriver to selenium folder and open our Laravel project in Selenium folder by Visual Studio Code.

Test Case:

Test Case for Login Module :

Project Name: E Commerce Site		Test Designed by: Aurthy		
Test Case ID: Login_01		Test Designed date: 17-07-2022		
Test Priority (Low, Medium, High): High		Test Executed by: Ananya		
Module Name: Login Session		Test Execution date: 18-07-2022		
Test Title: verify login with valid username and password				
Description: Test website login page				
Precondition (If any): User must have valid username and password				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website 2. Click “Login” button 3. Enter username 4. Enter password 5. Click Login	Username: Ayon07 Password: 123456	User should login into the application	As expected	Pass
Post Condition: User is validated with database and successfully login to account. The account session details are logged in the database.				

Test Case for User create Module :

Project Name: E Commerce Site		Test Designed by: Aurthy		
Test Case ID: create_user_001		Test Designed date: 17-07-2022		
Test Priority (Low, Medium, High): High		Test Executed by: Ananya		
Module Name: Creating user		Test Execution date: 18-07-2022		
Test Title: Create user for the website				
Description: Create User for the website for business purpose.				
Precondition (If any): 1. Need to be logged in on the website				
Test Steps	Test Data	Expected Results	Actual Results	Status (Pass/Fail)
1. Go to the website 2. Click “Login” button 3. Enter username 4. Enter password 5. Click Login 6. Click Create User 7. Enter User Name 8. Click submit	User Name: Ayon	User created successfully	As expected	Pass
Post Condition: Admin redirect to the dashboard page with a success message.				

Testing Process:

- **Login Module:**

After installing all the necessary files we log in to the server to open our project. After getting the IP-link we connected to the software that we asked the automated software to search for “username ” and write “Ayon07”. If the user name input is done correctly, we will get a message as user name input completed. The same goes to the password. In password field we will insert “123456” then press the button “Login”. If password works properly then we will get the answer as succussed or failed.

```
login.py > ...
1  from selenium.webdriver.common.by import By
2  from selenium import webdriver
3  import time
4  from selenium.webdriver.chrome.service import Service
5  from webdriver_manager.chrome import ChromeDriverManager
6  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
7  # driver = webdriver.Chrome(executable_path="chromedriver.exe")
8  driver.get("http://127.0.0.1:8000/login") ← Laravel project hyper-link
9  time.sleep(2)
10 #name = u_username
11 name = driver.find_element(By.ID, "u_username")
12 name.send_keys("Ayon07")
13 time.sleep(2)
14 print("Username input completed")
15 password = driver.find_element(By.ID, "u_password")
16 password.send_keys("123456")
17 print("Password input completed")
18 time.sleep(2)
19 #password = u_password
20 # id = login
21
22 button = driver.find_element(By.ID, "login")
23 button.click()
24 print("Login button cliked")
25
26 if driver.title == "User Dash":
27     print("Successfully Logged In")
28 else:
29     print("Login Failed")
30 time.sleep(2)
31 driver.quit()
32
```

Figure 3: Testing Login Module

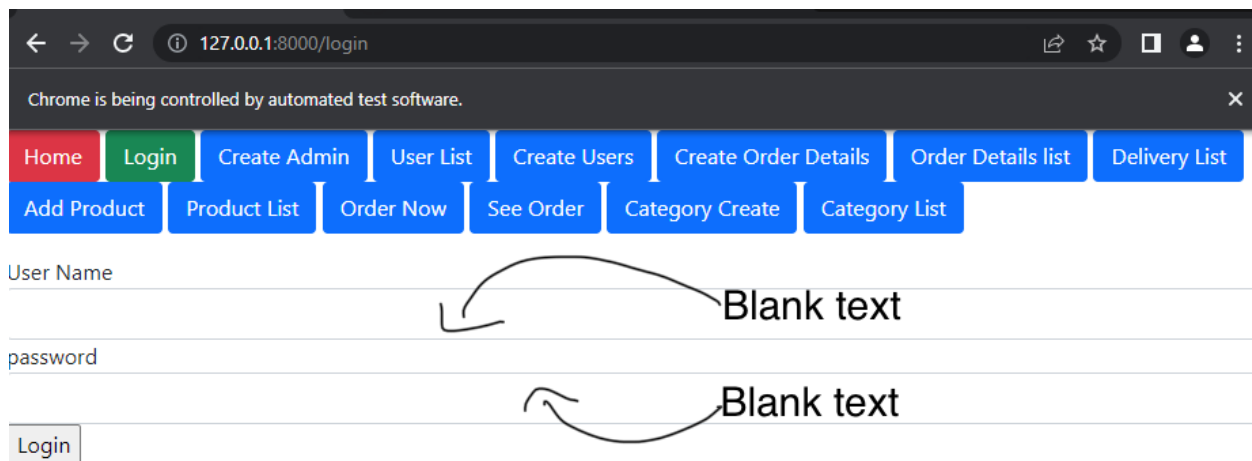


Figure 4: Input field for Login

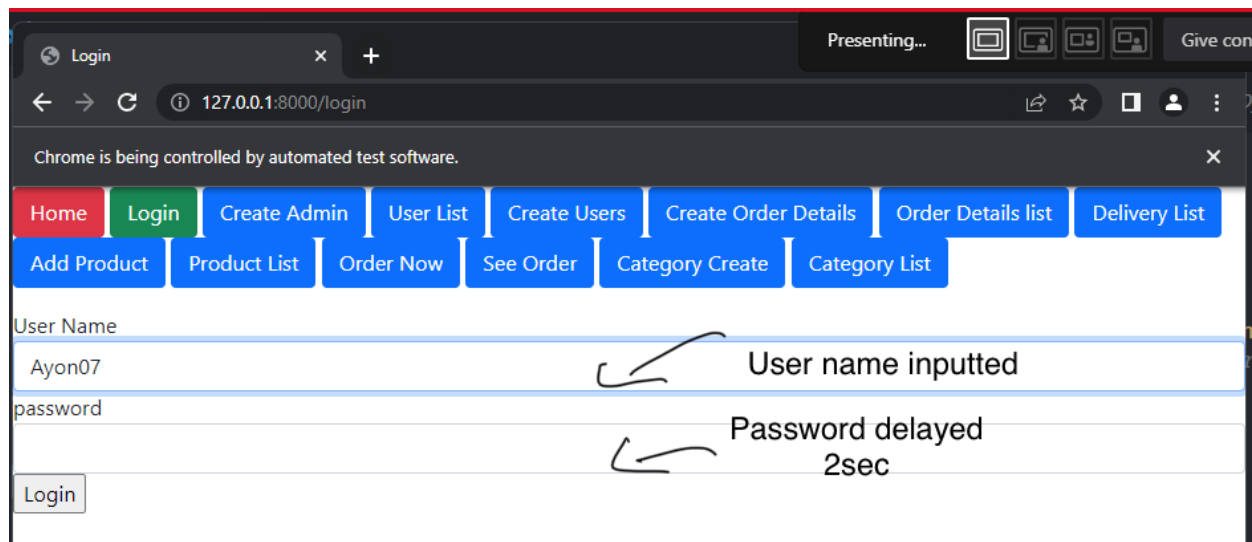


Figure 5: Insert user name

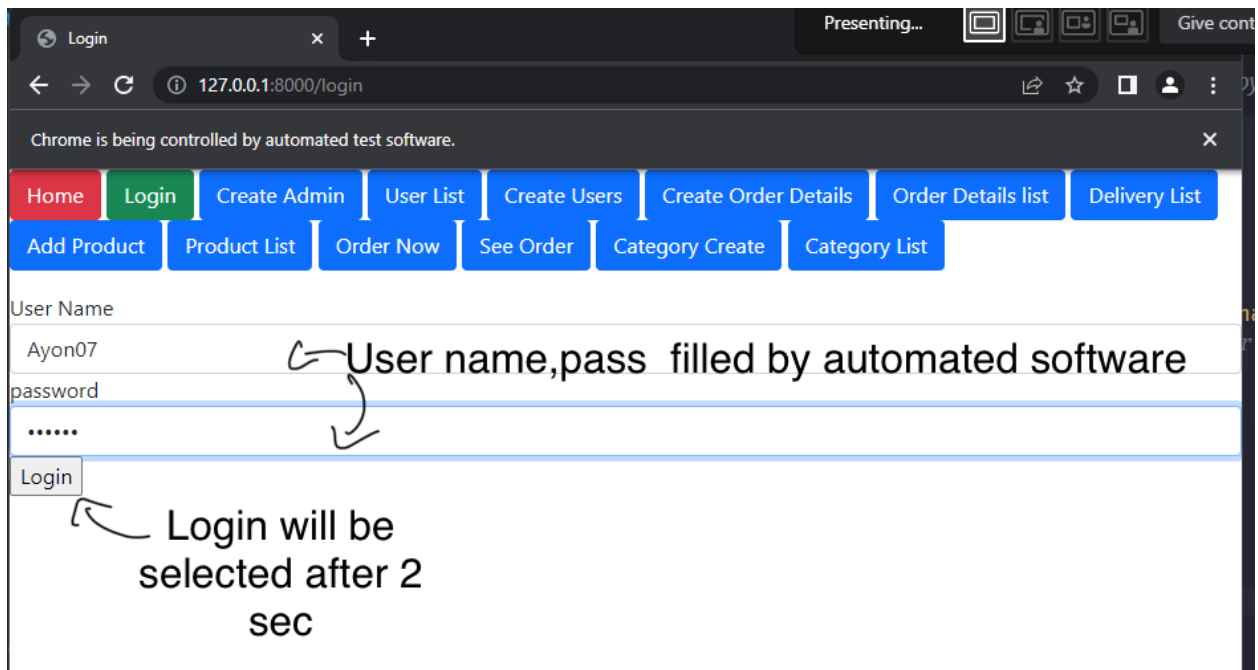


Figure 6: Insert Password

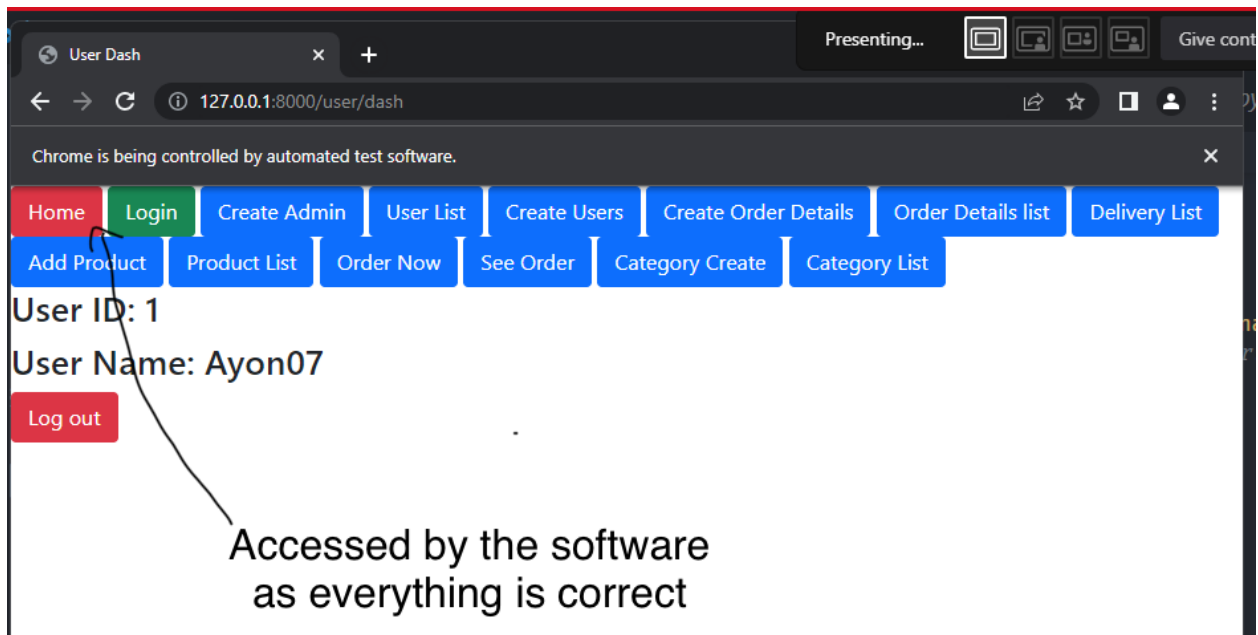


Figure 7: Successfully Login

```
PS D:\12thsem\sqt\selenium> py login.py

DevTools listening on ws://127.0.0.1:64935/devtools/browser/0b987e6f-3dad-496b-9ece-681878cd312b
[23800:23868:0718/204432.904:ERROR:device_event_log_impl.cc(214)] [20:44:32.903] Bluetooth: bluetooth_adapter_w
inrt.cc:1074 Getting Default Adapter failed.
Username input completed
Password input completed
Login button clicked
Successfully Logged In ← Accessed
```

Figure 8: Result of Login

In Figure 3, we have shown the code of login page testing. And after that when we execute the code at first we can see the login page of our ecommerce site and all the input fields are blank there in Figure 4. Also in Figure 5, we can see the username is automatically inserted. After that it will insert the Password which is in Figure 6. Then the login button will be clicked and it will enter the user dash board and we can see the Name of the user in Figure 7. And in Figure 8, we can see the output of the result either the input was given perfectly or not and the login was successful or not.

Result: As we inserted correct username and password provided by the teams and as it is logged in successfully. So we can say that the code was correctly implemented.

- **Error checking on Login Module:**

As correct password and username leads us to get the access of login to the software it's time to check whether the code can distinguish correct input from the wrong input and whether it grant us the access to log in.

```
from selenium.webdriver.common.by import By
from selenium import webdriver
import time
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
# driver = webdriver.Chrome(executable_path="chromedriver.exe")
driver.get("http://127.0.0.1:8000/login")
time.sleep(2)
#name = u_username
name = driver.find_element(By.ID, "u_username")
name.send_keys("Ayon07")
time.sleep(2)
print("Username input completed")
password = driver.find_element(By.ID, "u_password")
password.send_keys("1234567")
print("Password input completed")
time.sleep(2)
#password = u_password
# id = login

button = driver.find_element(By.ID, "login")
button.click()
print("Login button clicked")

if driver.title == "User Dash":
    print("Successfully Logged In")
else:
    print("Login Failed")
time.sleep(2)
driver.quit()
```

The diagram illustrates the error checking logic in the provided Python code. It features three annotations with arrows pointing to specific lines of code:

- Correct:** An arrow points from this label to the line `name.send_keys("Ayon07")`, indicating that this is the correct username input.
- Wrong input provided:** An arrow points from this label to the line `password.send_keys("1234567")`, indicating that this is an incorrect password input.
- Message:** An arrow points from this label to the `print("Login Failed")` line within the `else:` block, indicating the message output when the login fails.

Figure 9: Error checking

```
PS D:\12thsem\sgt\selenium> py login.py

DevTools listening on ws://127.0.0.1:63657/devtools/browser/4f46e8eb-cdc6-4e50-b300-ff494ad1847a
[14592:25216:0718/203508.636:ERROR:device_event_log_impl.cc(214)] [20:35:08.635] Bluetooth: bluetooth_adapter_winrt.cc:1074 Getti
ng Default Adapter failed.
Username input completed ✓
Password input completed ✓
Login button clicked ✓
Login Failed ← Denied access
```

Figure 10: Login failed as wrong password inserted

Result: We insert a wrong password over here for checking if our code is working properly or not. As we have given the wrong password in password input field in Figure 9, we can see that when all the insertion is done the Login was failed and in the output which is in Figure 10 we can see it printed Login Failed.


- **Adding User Module:**

In this test case we will creating an user automatically using Selenium. For this we will be providing username, password, phone number, Email and if every given data matched with the validation state the program will let us to create a new user.

```
createuser.py > ...
1  from selenium.webdriver.common.by import By
2  from selenium import webdriver
3  import time
4  from selenium.webdriver.chrome.service import Service
5  from webdriver_manager.chrome import ChromeDriverManager
6  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
7  # driver = webdriver.Chrome(executable_path="chromedriver.exe")
8  driver.get("http://127.0.0.1:8000/admin/create") ← Admin creation link
9  time.sleep(5)
10
11  name = driver.find_element(By.ID, "u_name")
12  name.send_keys("Ananyo Chowdhury") ←
13
14  phone = driver.find_element(By.ID, "u_phone")
15  phone.send_keys("01649889794") ←
16
17  address = driver.find_element(By.ID, "u_address")
18  address.send_keys("Gulshan, Dhaka") ←
19
20  username = driver.find_element(By.ID, "u_username")
21  username.send_keys("Ananyo10") ←
22
23
24  email = driver.find_element(By.ID, "u_email")
25  email.send_keys("Ananyo4@gmail.com") ←
26
27
28  password = driver.find_element(By.ID, "u_password")
29  password.send_keys("123456") ←
30
31  usertype = driver.find_element(By.ID, "Admin")
32  usertype.click()
```

Proving data to the automated software

```
31  usertype = driver.find_element(By.ID, "Admin")
32  usertype.click()
33
34  img = driver.find_element(By.ID, "u_profileimg")
35  img.send_keys("default.jpg")
36
37  button = driver.find_element(By.ID, "signup")
38  button.click()
39  print("Signup clicked")
40
41
42  if driver.title == "User List":
43      print("Successfully User Added")
44  else:
45      print("User Adding Failed")
46
47  time.sleep(5)
48  driver.quit()
49
```



Message

Fig 11: Creating User testing Code

Admin page x +

127.0.0.1:8000/admin/create

Chrome is being controlled by automated test software.

Home Login Create Admin User List Create Users Create Order Details Order Details list Delivery List

Add Product Product List Order Now See Order Category Create Category List

Name

Input blank

Phone

Input blank

Address

Input blank

Username

Input blank

Email

Input blank

Password

Input blank

User Type: Select Unselected

Profile Picture

default.jpg

Sign Up Log out

Figure 12: Create User input field

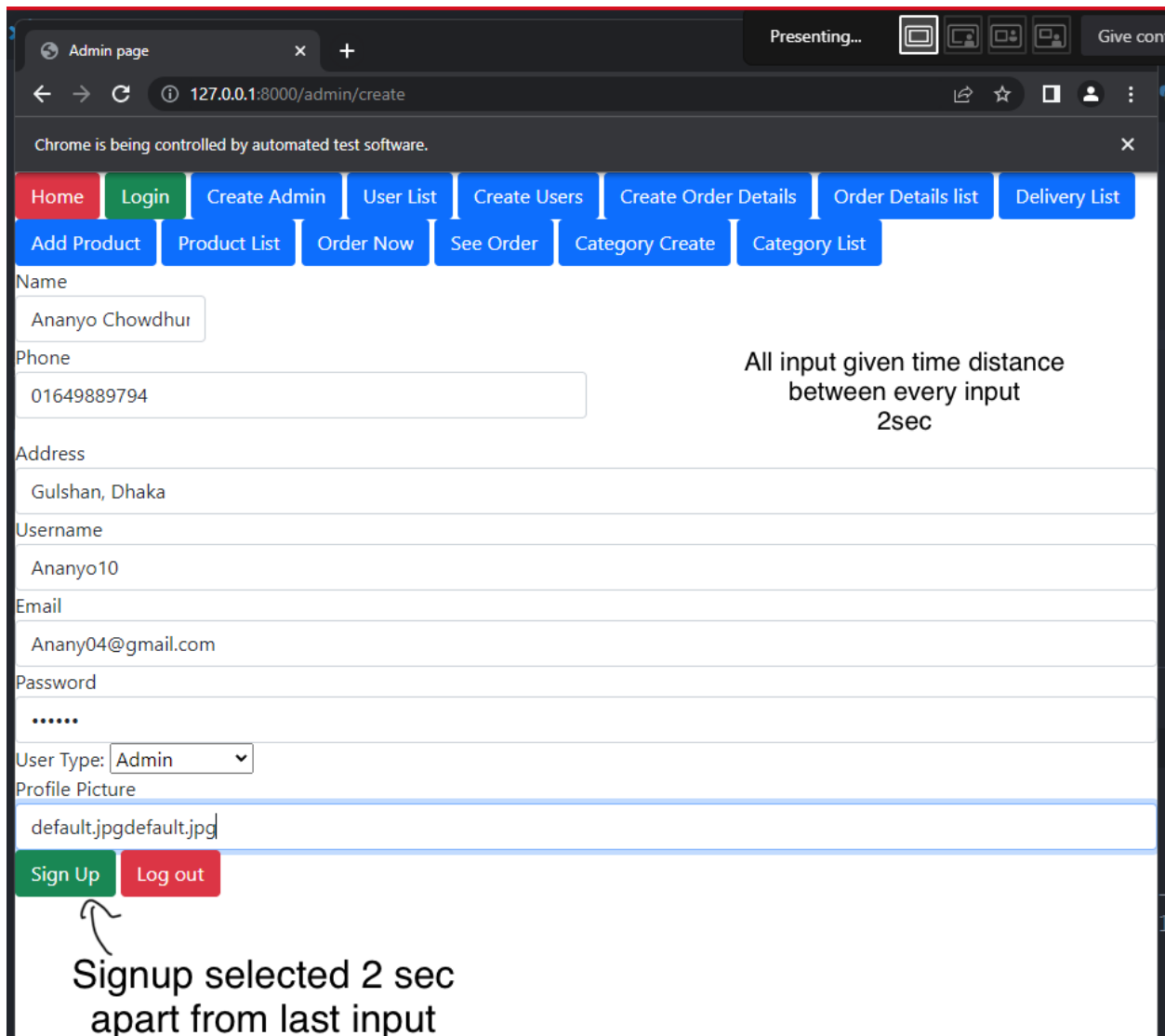


Figure 13: All the input field filled automatically

```
PS D:\12thsem\sgt\selenium> py createuser.py

DevTools listening on ws://127.0.0.1:52565/devtools/browser/c0c8ab46-6867-47bc-94ab-336f9d13b1be
[9664:12484:0718/212156.124:ERROR:device_event_log_impl.cc(214)] [21:21:56.124] Bluetooth: bluetooth_adapter_winrt.cc:1074 Getting Default Adapter failed.
Signup clicked
Successfully User Added
PS D:\12thsem\sgt\selenium>
```

Figure 14: output of the create-user test

Here, On Figure 11, we can see the testing code of creating new user. Then when we execute the code, the chrome file opens the creating user link in Figure 12 where the input field is blank and after that in the Figure 13, the input field was filled automatically. And that Figure 14, output of the test file and user was added successfully.

Result: As we inserted correct and validate inputs we can see that the user was added successfully and it printed "Successfully User Added".

- **Counting Module:**

Here we count all the Anchor tags from the Dashboard and show those sequentially by this code. We have in total 11 Anchor tags in Dashboard.

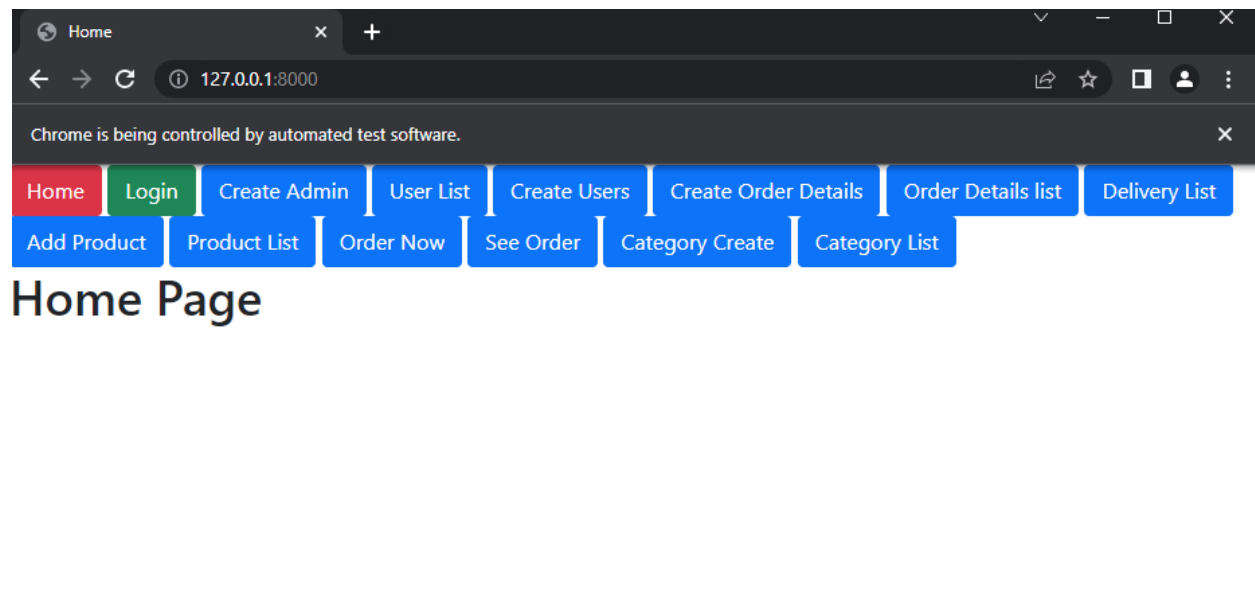


Figure15 :All anchor tags from Dashboard

```

ok.py | links.py | implicit.py | handle.py | login.py
links.py > ...
1  import time
2  from selenium import webdriver
3  from selenium.webdriver.common.by import By
4  from selenium.webdriver.chrome.service import Service
5  from webdriver_manager.chrome import ChromeDriverManager
6  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
7
8
9  # driver = webdriver.Chrome(executable_path="chromedriver.exe")
10
11  driver.get("http://127.0.0.1:8000")
12
13  links = driver.find_elements(By.TAG_NAME, "a")
14  print("Length of anchor tag: ", len(links))
15
16
17  # link TEXT
18  for link in links:
19      print(link.text)
20
21  time.sleep(4)
22
23  driver.find_element(By.LINK_TEXT, "Login").click()
24  time.sleep(4)
25
26  driver.quit()
27

```

Figure16 : Counting Module

```

Length of anchor tag: 14      Number of anchor tag
Home
Login
Create Admin
User List
Create Users
Create Order Details
Order Details list      Name of all anchor tag
Delivery List
Add Product
Product List
Order Now
See Order
Category Create
Category List

```

Figure17 : Length and list of all anchor tag name

Result: We supposed to get the length of anchor tags along with the names of all anchor tag and we got the actual result.

- **Multitasking handle Module:**

In this test case we are selecting the 5th element on the top navigation bar and moved towards that page and click that button , we can see its on create users page. In this program we are trying to open home from the top navigation . we are selecting this from the title drive. And stay on this case for 5 sec. To opened another page from the navigation bar that is create user . we can see the tin the page that it opened another page on the chrome bar. In this case we are checking we there is any index title name as home if we found home in the bar we will close that page and keep other pages turned open.

```
handle.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  import time
4  from selenium.webdriver.chrome.service import Service
5  from webdriver_manager.chrome import ChromeDriverManager
6  driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
7
8  driver.get("http://127.0.0.1:8000")
9
10
11  time.sleep(4)
12
13  var = driver.find_element(by=By.XPATH, value="/html/body/a[5]")
14  print(driver.current_window_handle)
15
16  time.sleep(4)
17  # current_window_handle == parent
18
19  handles = driver.window_handles
20  # window_handles -- child handle
21
22  for handle in handles:
23      driver.switch_to.window(handle)
24      # print(driver.title)
25
26      if driver.title == "Home":
27          driver.close()
28          print("Home Page closed")
29
30  time.sleep(5)
31
32  driver.quit()
33
```

Fig 18: Multipage Handler

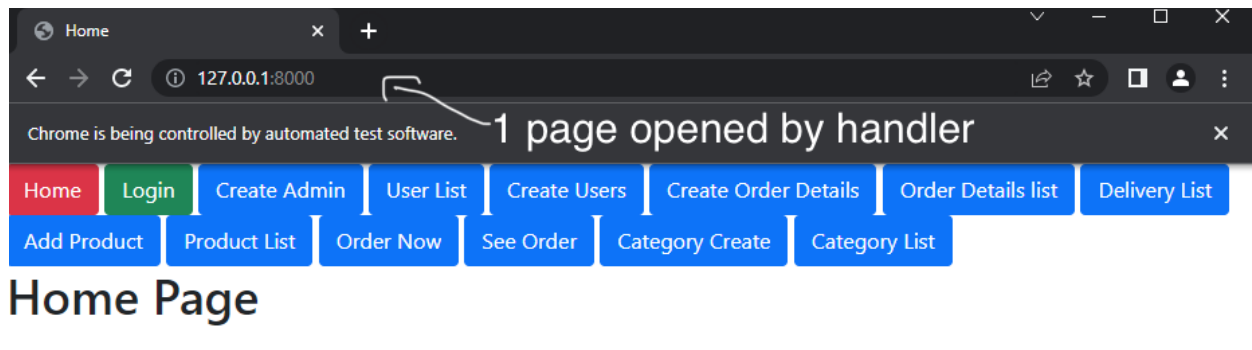


Fig 19: Multipage Handler

In the fig:19 by the help of handle function one page is opened in the chrome named as home anchor tag.as there was is a delay on 4 sec . For 4 second page will remain like this as like code.

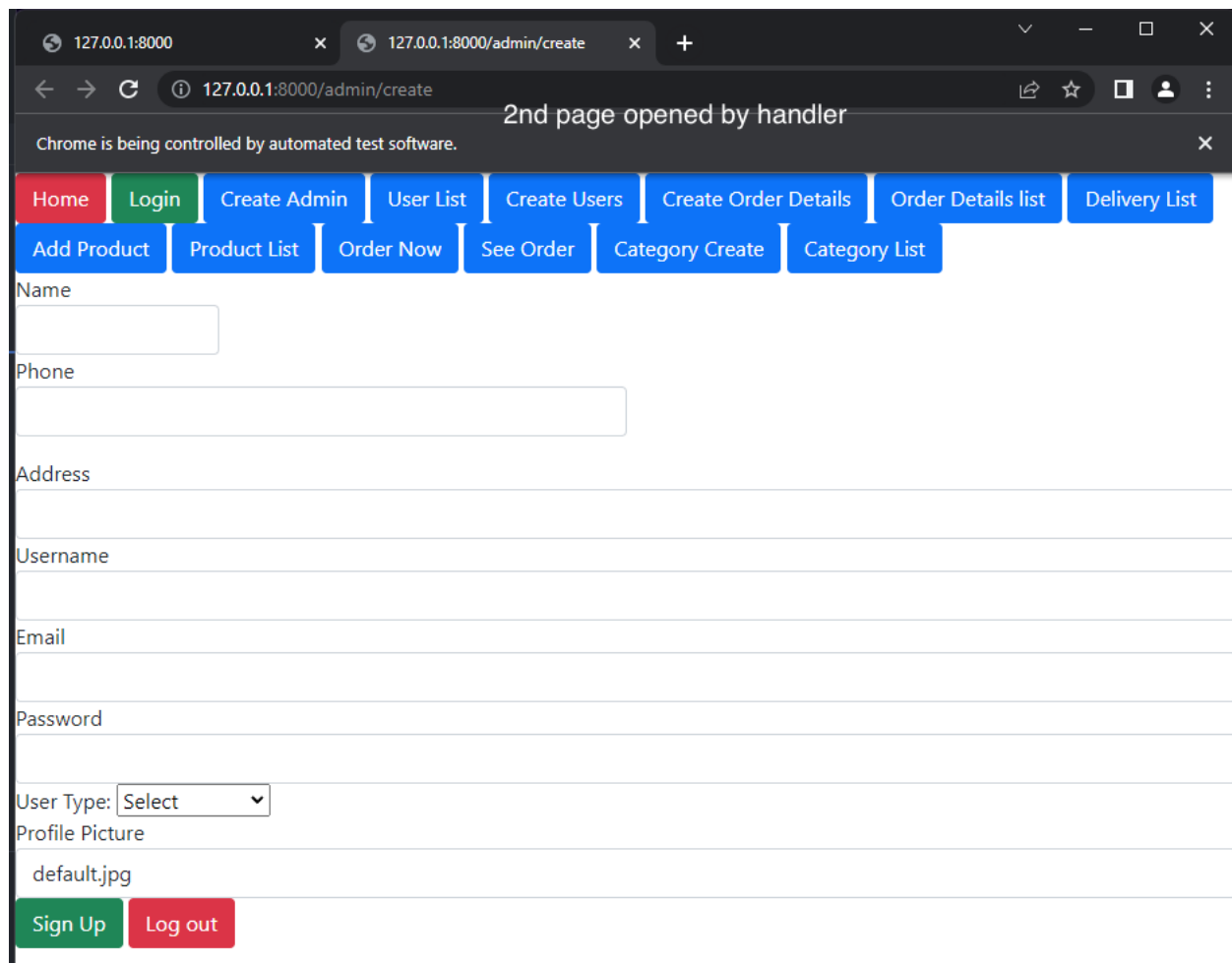
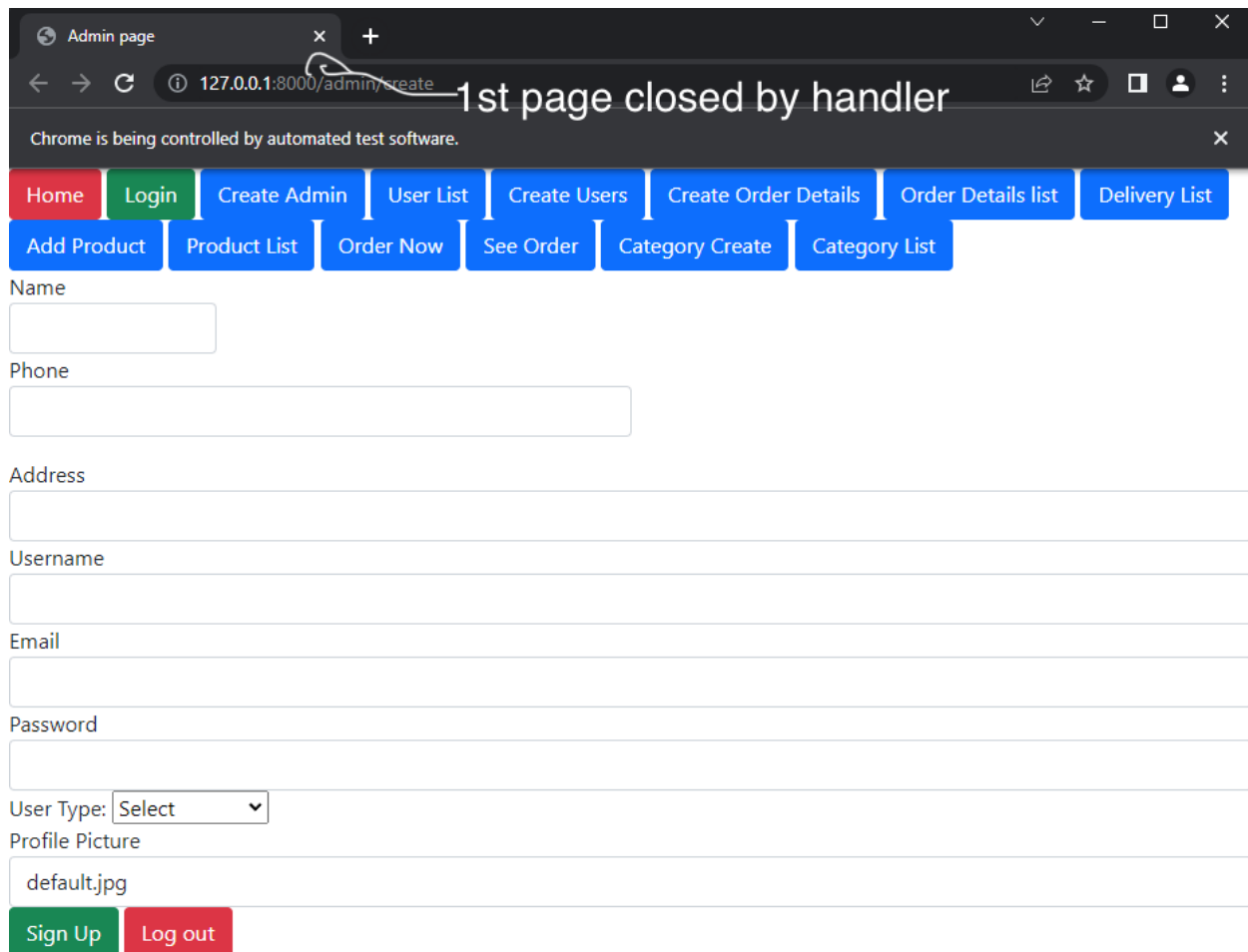


Fig 20: Multipage Handler

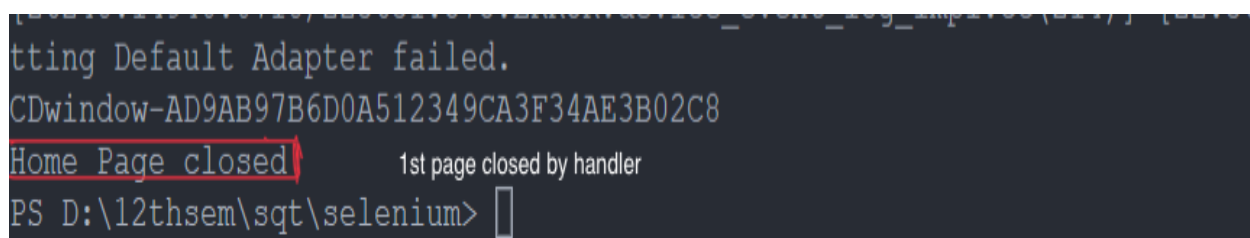
In fig:20 , 2nd page is opened by the help of handle function which is done by automated software , this page will remain open for 4sec after that next function will be called and be executed.



The screenshot shows a web browser window titled 'Admin page'. The address bar displays '127.0.0.1:8000/admin/create'. A text overlay '1st page closed by handler' points to the browser tab. The page contains a navigation bar with buttons: Home, Login, Create Admin, User List, Create Users, Create Order Details, Order Details list, Delivery List, Add Product, Product List, Order Now, See Order, Category Create, and Category List. Below the navigation bar is a form with fields for Name, Phone, Address, Username, Email, Password, User Type (a dropdown menu), and Profile Picture (with 'default.jpg' entered). At the bottom are 'Sign Up' and 'Log out' buttons.

Fig 21: Multipage Handler

In Fig:21 1st page is closed as in the code it was searching by the name home and as it finds that index page it will turn that page off which happens in this fig 23.



```
...tting Default Adapter failed.  
CDwindow-AD9AB97B6D0A512349CA3F34AE3B02C8  
Home Page closed 1st page closed by handler  
PS D:\12thsem\sqt\selenium>
```

Fig 22: Multipage Handler

In the fig:22 we can see the output generated by the automated software page it was designated by is closed.

