**Experiment No. - 10**

**Name-** Bhavesh Kewalramani

**Roll No.-** A-25

**Section-**  A

**Semester-** 6th

**Shift-** 1st

**Aim:**

To demonstrate web application testing using Selenium Web Driver.

**Hardware Required:** Windows/Linux/Mac Laptop or Desktop with minimum 4GB RAM.

**Software Required:**

1. Drivers
   1. Chrome Driver – [Take Me to Download Page](https://chromedriver.storage.googleapis.com/index.html)
   2. Mozilla -GeckoDriver – [Take Me to Download Page](https://github.com/mozilla/geckodriver/releases/)
   3. Internet Explorer Driver –[Take Me to Download Page](https://www.seleniumhq.org/download/)
2. Web Browsers
   1. Chrome – <https://www.google.com/chrome/>
   2. Internet Explorer – <https://www.microsoft.com/en-gb/download/>
   3. Firefox – [https://www.mozilla.org](https://www.mozilla.org/)
   4. Safari – Comes with Mac OS package.

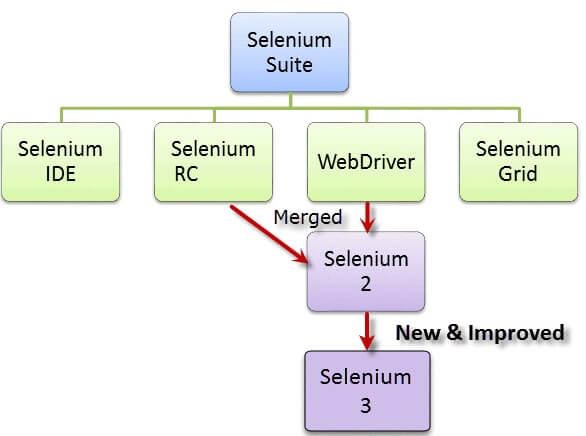
**Theory:**

## What is Selenium?

**Selenium** is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization. Here is the list of tools

* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid

Introduction to Selenium

At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**. Selenium 1, by the way, refers to Selenium RC.

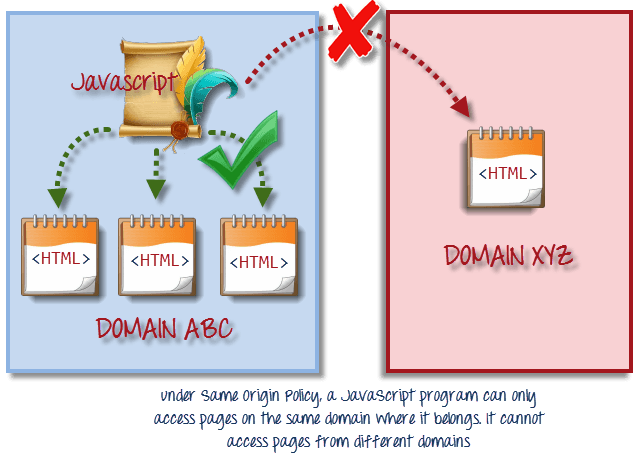
## Who developed Selenium?

Since Selenium is a collection of different tools, it had different developers as well. Below are the key persons who made notable contributions to the Selenium Project

|  |  |
| --- | --- |
| Introduction to Selenium | Primarily, Selenium was **created by Jason Huggins in 2004**. An engineer at ThoughtWorks, he was working on a web application that required frequent testing. Having realized that the repetitious [Manual Testing](https://www.guru99.com/manual-testing.html) of their application was becoming more and more inefficient, he created a[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)program that would automatically control the browser’s actions. He named this program as the “**JavaScriptTestRunner**.”  Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**. |

## The Same Origin Policy Issue

**Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched**. Example, the HTML code in www.google.com uses a JavaScript program “randomScript.js”. The same origin policy will only allow randomScript.js to access pages within google.com such as google.com/mail, google.com/login, or google.com/signup. However, it cannot access pages from different sites such as yahoo.com/search or guru99.com because they belong to different domains.



This is the reason why prior to Selenium RC, testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain

## Birth of Selenium Remote Control (Selenium RC)



Unfortunately; testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy.**So another ThoughtWork’s engineer, **Paul Hammant**, decided to create a server that will act as an HTTP proxy to “trick” the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.

## Birth of Selenium Grid



Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system “**Hosted QA**.” It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously.**

## Birth of Selenium IDE

****

**Shinya Kasatani**of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.

## Birth of WebDriver

****

**Simon Stewart**created WebDriver circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. **It was the first cross-platform testing framework that could control the browser from the OS level.**

## Birth of Selenium 2

In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project’s efforts are now focused on Selenium 2.

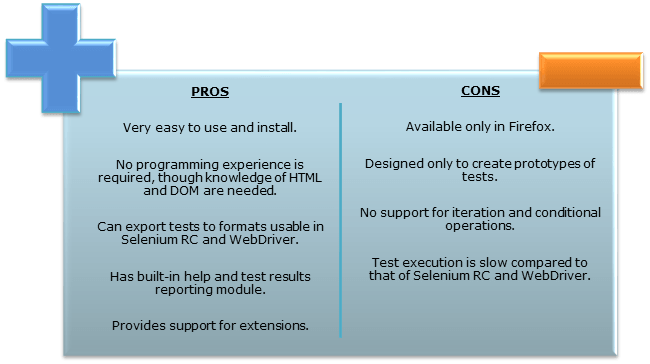
## So, Why the Name Selenium?

**The Name Selenium** came from a joke which Jason cracked once to his team. During Selenium’s development, another [automated testing](https://www.guru99.com/automation-testing.html) framework was popular made by the company called **Mercury Interactive**(yes, the company who originally made QTP before it was acquired by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name and his teammates took it. So that is how we got to call this framework up to the present.



## Brief Introduction Selenium IDE

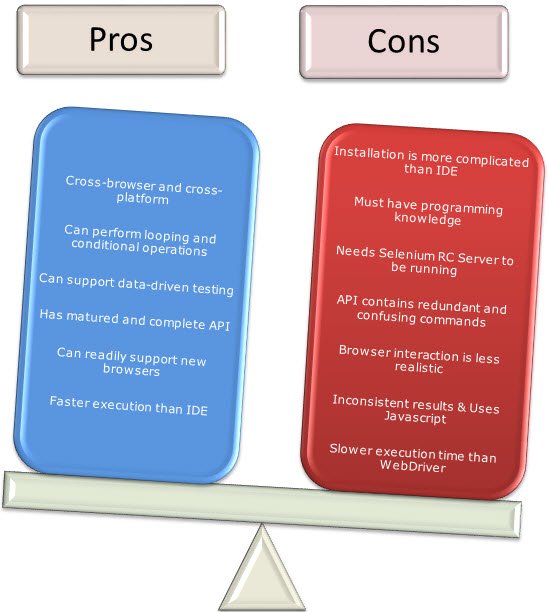
Selenium Integrated Development Environment (IDE) is the **simplest framework** in the Selenium suite and is **the easiest one to learn**. It is a **Firefox plugin** that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a **prototyping tool**. If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver.



## Brief Introduction Selenium Remote Control (Selenium RC)

Selenium RC was the **flagship testing framework** of the whole Selenium project for a long time. This is the first automated [web testing](https://www.guru99.com/web-application-testing.html) tool that **allowed users to use a programming language they prefer**. As of version 2.25.0, RC can support the following programming languages:

* [Java](https://www.guru99.com/java-tutorial.html)
* [C#](https://www.guru99.com/c-sharp-tutorial.html)
* [PHP](https://www.guru99.com/php-tutorials.html)
* Python
* Perl
* Ruby

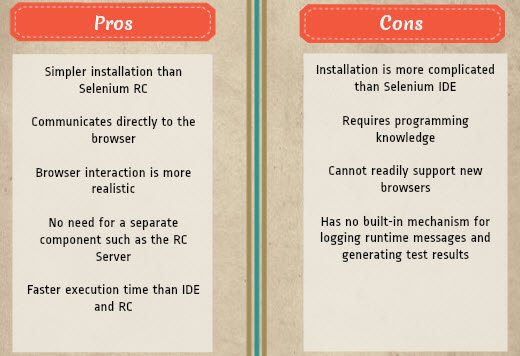


## Brief Introduction WebDriver

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser’s actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. **It controls the browser by directly communicating with it.**

The supported languages are the same as those in Selenium RC.

* Java
* C#
* PHP
* Python
* Perl
* Ruby



## Selenium Grid

Selenium Grid is a tool **used together with Selenium RC to run**[**parallel tests**](https://www.guru99.com/parallel-testing.html) across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once.

**Features:**

* Enables **simultaneous running of tests** in **multiple browsers and environments.**
* **Saves time**enormously.
* Utilizes the **hub-and-nodes** concept. The hub acts as a central source of Selenium commands to each node connected to it.

## Note on Browser and Environment Support

Because of their architectural differences, Selenium IDE, Selenium RC, and WebDriver support different sets of browsers and operating environments.

|  | **Selenium IDE** | **WebDriver** |
| --- | --- | --- |
| **Browser Support** | Mozilla Firefox | Internet Explorer versions 6 to 11, both 32 and 64-bit  Microsoft Edge version 12.10240 & above ( partial support some functionalities under development)  Firefox 3.0 and above  Google Chrome 12.0. and above  Opera 11.5 and above  Android – 2.3 and above for phones and tablets (devices & emulators)  iOS 3+ for phones (devices & emulators) and 3.2+ for tablets (devices & emulators)  HtmlUnit 2.9 and above |
| **Operating System** | Windows, Mac OS X, Linux | All operating systems where the browsers above can run. |

**Note:** Selenium WebDriver is termed as the successor of Selenium RC which has been deprecated & officially announced by SeleniumHQ.

## How to Choose the Right Selenium Tool for Your Need

| **Tool** | **Why Choose?** |
| --- | --- |
| **Selenium IDE** | * To learn about concepts on automated testing and Selenium, including: * Selenese commands such as type, open, clickAndWait, assert, verify, etc. * Locators such as id, name, xpath, css selector, etc. * Executing customized JavaScript code using runScript * Exporting test cases in various formats. * To create tests with little or no prior knowledge in programming. * To create simple test cases and test suites that you can export later to RC or WebDriver. * To test a web application against Firefox and Chrome only. |
| **Selenium RC** | * To design a test using a more expressive language than Selenese * To run your test against different browsers (except HtmlUnit) on different operating systems. * To deploy your tests across multiple environments using Selenium Grid. * To test your application against a new browser that supports JavaScript. * To test web applications with complex AJAX-based scenarios. |
| **WebDriver** | * To use a certain programming language in designing your test case. * To test applications that are rich in AJAX-based functionalities. * To execute tests on the HtmlUnit browser. * To create customized test results. |
| **Selenium Grid** | * To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. * To run a huge test suite, that needs to complete in the soonest time possible. |

## A Comparison between Selenium and QTP(now UFT)

**Quick Test Professional(QTP)**is a proprietary automated testing tool previously owned by the company **Mercury Interactive** before it was **acquired by Hewlett-Packard in 2006**. The Selenium Tool Suite has many advantages over  QTP as detailed below –

Advantages and Benefits of Selenium over QTP

|  |  |
| --- | --- |
| **Selenium** | **QTP** |
| **Open source**, **free to use**, and **free of charge.** | **Commercial**. |
| **Highly extensible** | Limited add-ons |
| Can run tests across **different browsers** | Can only run tests in **Firefox**, **Internet Explorer** and **Chrome** |
| Supports **various operating systems** | Can only be used in **Windows** |
| Supports **mobile devices** | QTP Supports Mobile app test automation (iOS & Android) using HP solution called – HP Mobile Center |
| Can execute tests **while**the **browser is minimized** | Needs to have the application under test to be visible on the desktop |
| Can execute tests **in parallel**. | Can only execute in parallel but using Quality Center which is again a paid product. |

## Advantages of QTP over Selenium

|  |  |
| --- | --- |
| Advantages of QTP over Selenium | |
| **QTP** | **Selenium** |
| Can test **both web and desktop applications** | Can only test web applications |
| Comes with a **built-in object repository** | Has no built-in object repository |
| **Automates faster than Selenium** because it is a fully featured IDE. | Automates at a slower rate because it does not have a native IDE and only third party IDE can be used for development |
| Data-driven testing is easier to perform because **it has built-in global and local data tables**. | Data-driven testing is more cumbersome since you have to rely on the programming language’s capabilities for setting values for your test data |
| **Can access controls within the browser**(such as the Favorites bar, Address bar, Back and Forward buttons, etc.) | Cannot access elements outside of the web application under test |
| Provides professional **customer support** | No official user support is being offered. |
| Has native capability to **export test data** into external formats | Has no native capability to export runtime data onto external formats |
| Parameterization Support is built | Parameterization can be done via programming but is difficult to implement. |
| Test Reports are generated automatically | No native support to generate test /bug reports. |

Though clearly, QTP has more advanced capabilities, Selenium outweighs QTP in three main areas:

* **Cost**(because Selenium is completely free)
* **Flexibility**(because of a number of programming languages, browsers, and platforms it can support)
* **Parallel testing**(something that QTP is capable of but only with use of Quality Center)

## Summary

* The entire Selenium Software Testing Suite is comprised of four components:
* Selenium IDE, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.
* Selenium Remote Control, also known as Selenium 1, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.
* WebDriver, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.
* Selenium Grid is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.
* Selenium RC and WebDriver was merged to form Selenium 2.
* Selenium is more advantageous than QTP in terms of costs and flexibility. It also allows you to run tests in parallel, unlike in QTP where you are only allowed to run tests sequentially.

**Installing and working with selenium and webdrivers:**

### **Configure Selenium using Python**

There are following steps to configure Selenium using Python:

* **Download and install Python on Windows**
* **Install Selenium libraries in Python**
* **Download and install PyCharm**
* **Create a new project and write the Selenium test script**
* **Run and validate the test scripts.**

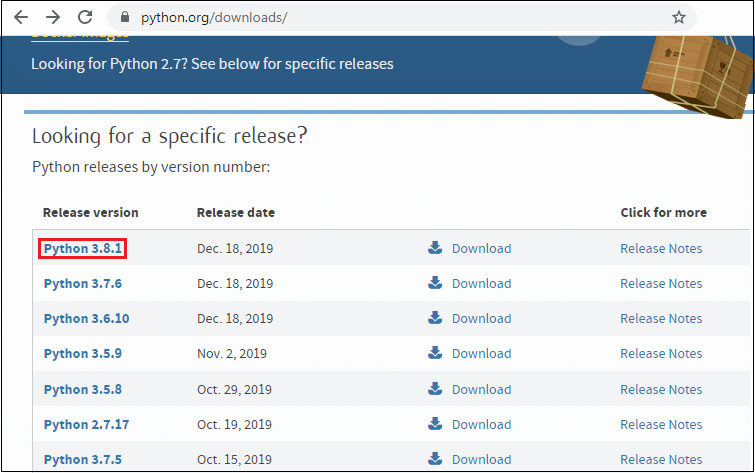
### **Download and install Python for Windows**

In this section, we will see how we download and install the Python for Windows platform.

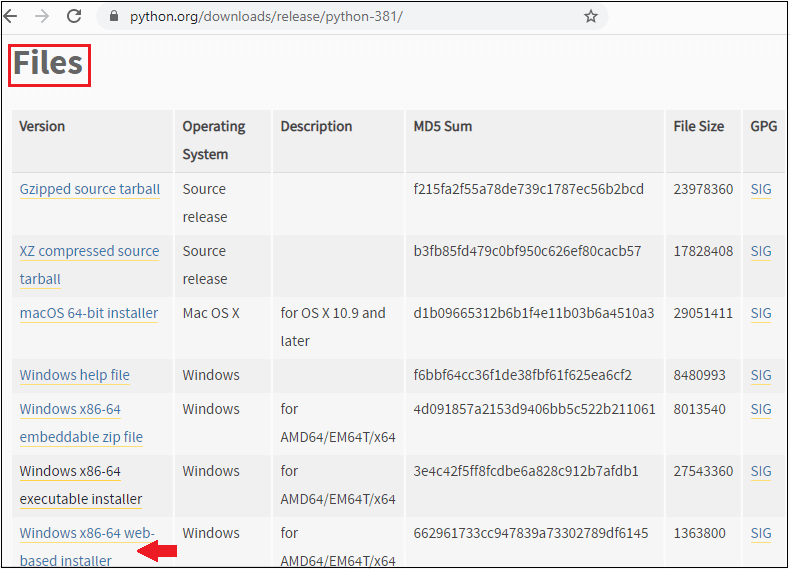
**Download the Python**

To download the latest version of Python for Windows Platforms, refer the below link: [https://www.Python.org/downloads/](https://www.python.org/downloads/)

* Once we clicked on the above link, the latest Release version list is shown, where we clicked on the **Python 3.8.1 version** as we can see in the below screenshot:



* The **Python-3.8.1** version window will appear on the screen, then scroll the page little-bit and find the**File** section, and the click on the **Windows x86-64 web-based installer** link for the Windows operating system as we can see in the below screenshot:

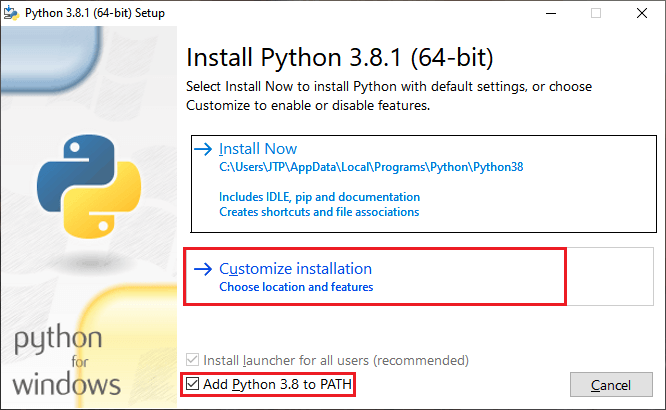


**Install the Python**

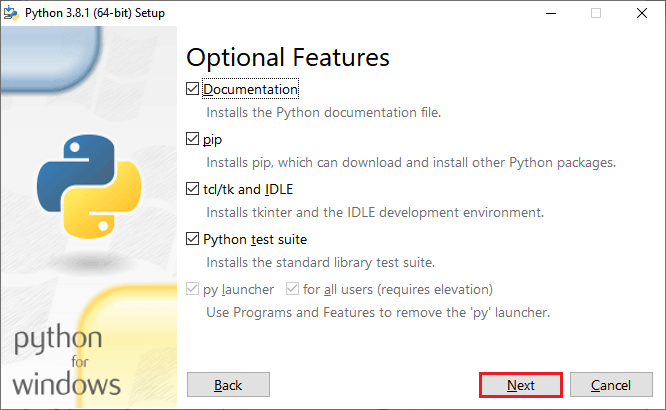
After downloading the Python for **Windows-64 bit**, we will be ready to install the Python.

To install the Python, follow the below process:

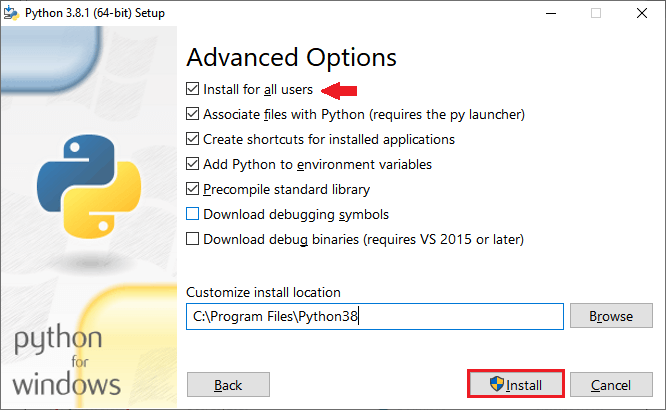
* Once we double-click on the downloaded executable file, the **Python 3.8.1(64-bit)** setup window will appear on the screen, where we have two options available to install the Python, which are:
  + **Install Now**
  + **Customize installation**
* We will click on the **Customize installation,** and select **Add Python 3.8 to path** checkbox as we can see in the below image:



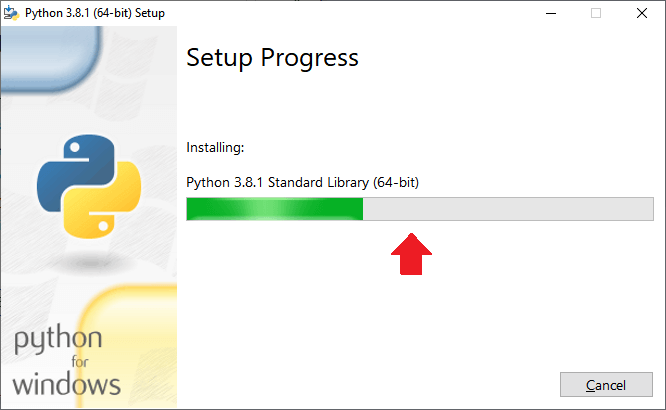
* After, click on the customize installation, the **Optional Features** will appear on the screen, where we can select and deselect the features according to our requirements.
* Then, click on the **Next** button, to proceed further as we can see in the below image:



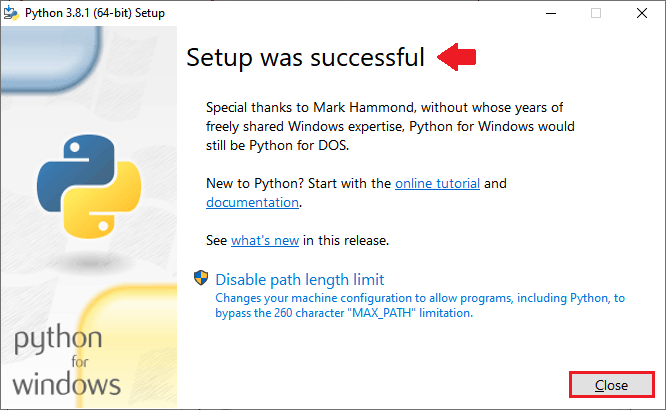
* Once, we clicked on the Next button; we have a list of **Advanced Options** available, where we can select the options based on our needs and also make sure that the **Install for all users** is selected.
* We can also customize the **install location** according to our convenience by clicking on the **Browse**
* After that, click on the **Install** button, to install the Python as we can see in the below screenshot:



* The installing process is getting started after clicking on the Install button as we can see in the below screenshot:

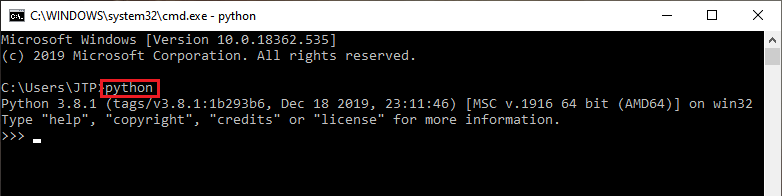


* When the installation is done, we got the confirmation message as **Setup was successful,**which means that the Python is installed successfully for the **Windows**operating system.
* Then, click on the **Close** button, to close the setup window as we can observe in the below screenshot:



After that, we will check whether Python is installed successfully and working fine or not.

So for this, we will open our command prompt, and type the command as **Python** and press the **Enter key**, and it will open the Python interpreter shell where we can implement the Python program as we can see in the below image:



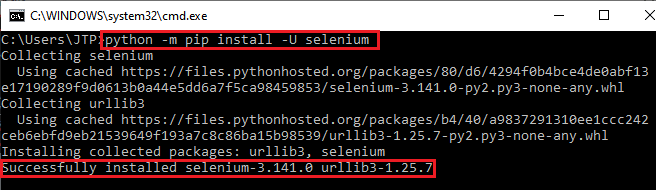
### **Installing the Selenium libraries in Python**

Once we successfully install the Python in our operation system, we will install the Selenium libraries.

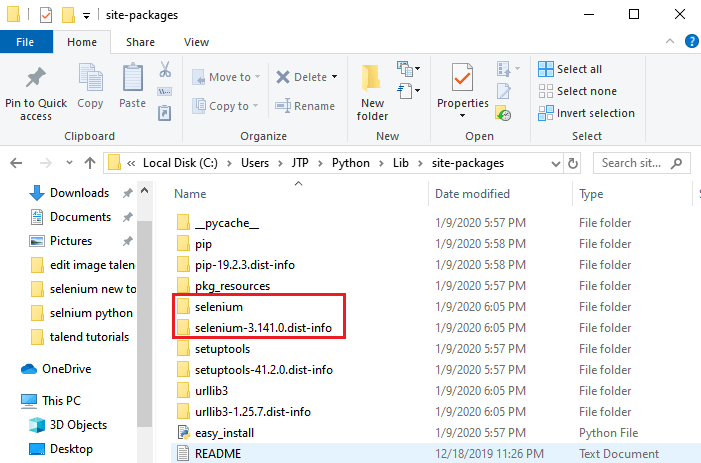
For this, we will execute the following command in our command prompt:

1. Python -m pip install -U Selenium

And, this command will successfully install the latest **Selenium package** i.e., **Selenium -3.141.0** added to the libraries as we can see in the below image:



After that executing the above command, it will create the **Selenium folder** automatically having all the Selenium libraries as we can see in the below screenshot:

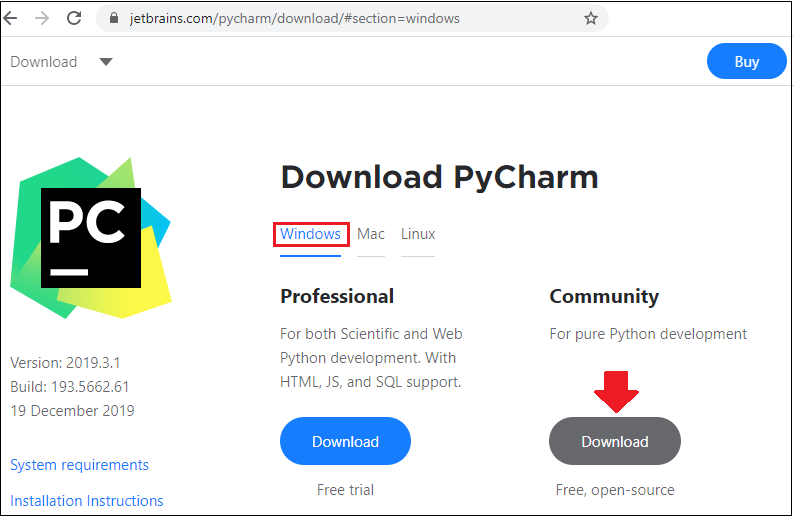


### **Download and install PyCharm**

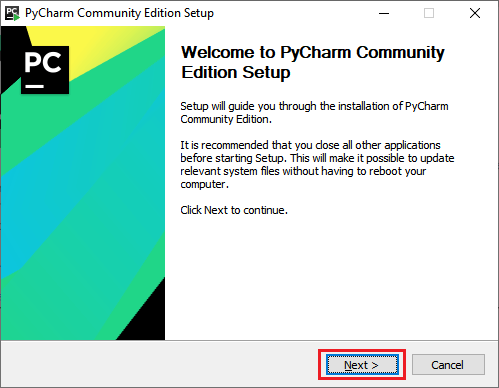
Once we successfully install the Selenium libraries into Python, we are ready to download Python IDE that is PyCharm.

To download the PyCharm, follow the below process:

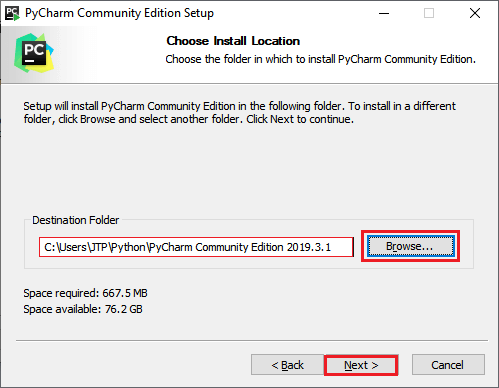
* Refer the below link, to download the PyCharm <https://www.jetbrains.com/pycharm/download/#section=windows>
* Once we clicked on the above link, we will get the below window, where will click on the**Download** button under the **Community**section for the **Windows**



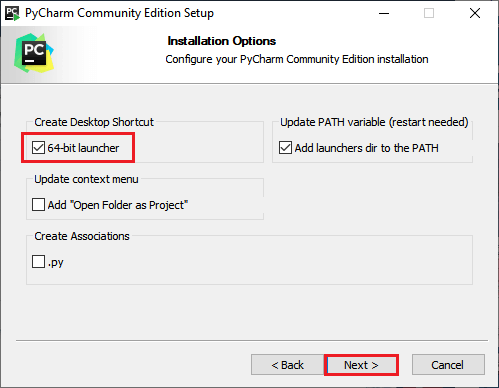
* After that, double-click on the executable file to install the PyCharm, and the **PyCharm Community Edition Setup** window will appear on the screen, where we click on the **Next** button to proceed further as we can see in the below image:



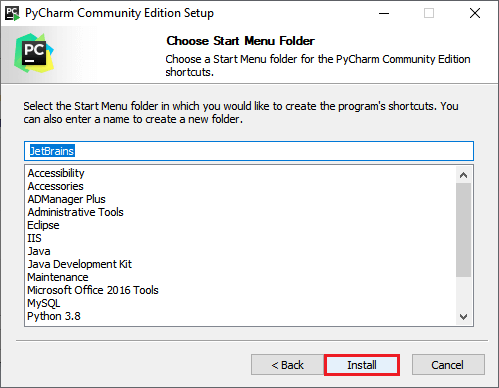
* In the next step, we can **Choose Install location** by clicking on the **Browser** button, then click on the **Next** button for further process.



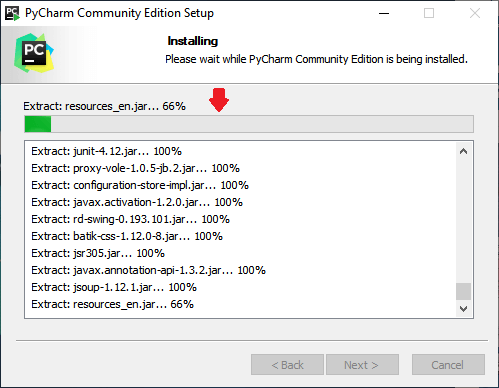
* In the next step, we have some **Installation Options** available, and we can select them based on our requirements.
* After that, click on the **Next** button as we can see in the below image:



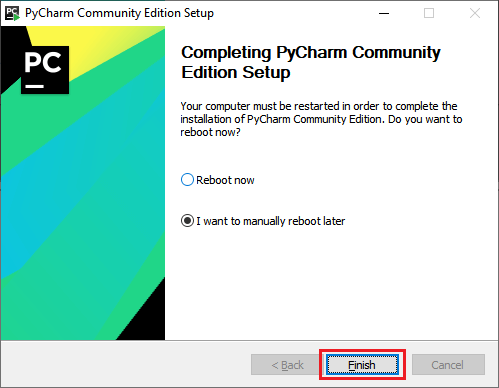
* Then, click on the **Install** button to install the PyCharm, as we can see in the below screenshot:



* As we can see in the below image, the installation process is getting started.



* Then, click on the **Finish** button to finish the installation process as we can see in the below image:



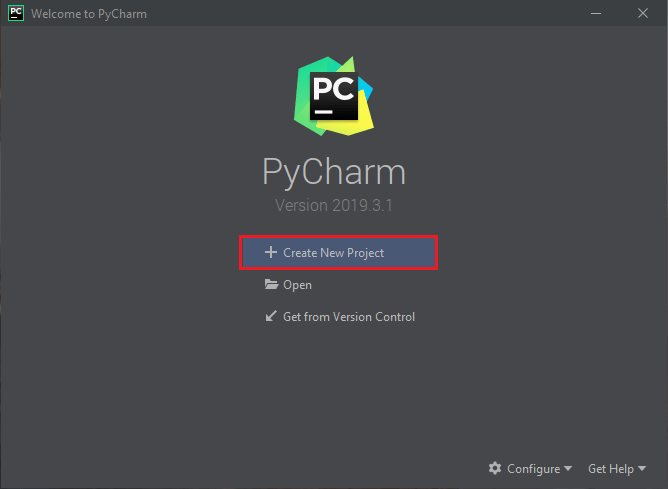
### **Create a new project and write the Selenium test script**

Once we successfully install the PyCharm, we will open the PyCharm IDE for creating a new project.

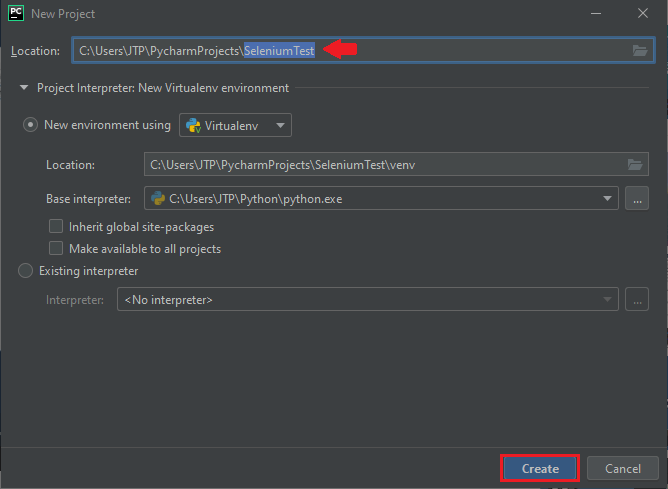
**Create a New Project in PyCharm**

Follow the below process, to create a new project in PyCharm:

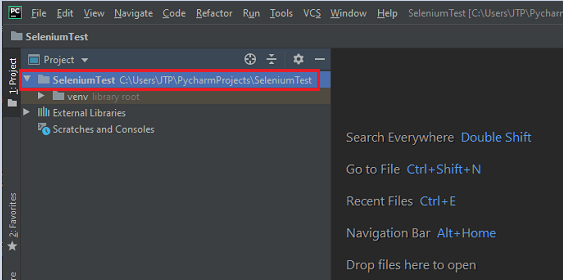
* First, open the PyCharm by Double-click on it, and click on the **Create New Project** as we can see in the below image:



* After that, we will provide the project name as **SeleniumTest**, and click on the **Create** button as we can see in the below image:



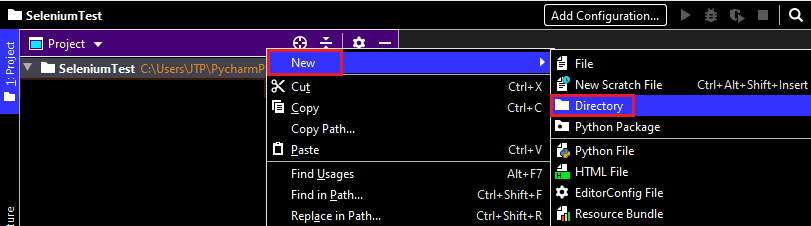
* After clicking on the Create button, we will get the below window:



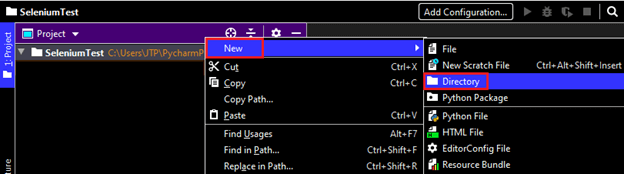
**Adding Selenium Test Scripts**

For adding the Selenium test scripts in the PyCharm, follow the below process:

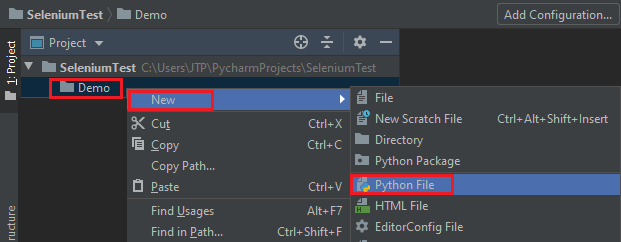
* Right-click on the **SeleniumTest** project, then go to **New,** and we can add any of the options in the given list according to our requirements.
* But, here we are adding the Python file, so for this, we will add the **Directory** which helps us to manage them separately as we can see in the below screenshot:



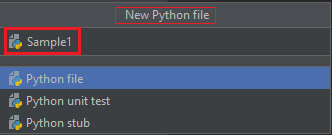
* And, provide the Directory name, in our case we give it as **Demo**
* After that, press the **Enter** key as we can see in the below screenshot:



* After creating a Directory, we will right-click on the **Demo**Directory then go to **New**, and select **Python File** from the pop-up menu as we can see in the below image:  
  **Demo → New → Python File**



* And, we provide a name to python file as **Sample1**.
* Then, press the **Enter** key as we can see in the below image:



* After that, we got the IDE where we can create or write our Selenium test Scripts.

**Write the Selenium test script**

For our testing purpose, we will first go to the **Google Home page** and search **javatpoint** from there.

We are creating our sample test script step by step to give you a complete understanding of how we write a Selenium test script in Python programming language.

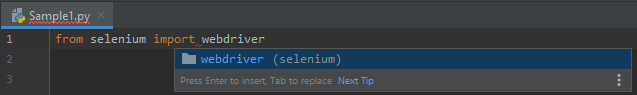
For this, follow the below steps:

|  |  |  |  |
| --- | --- | --- | --- |
| **Steps** | **Actions** | **Input** | **Expected Result** |
| **1.** | Import WebDriver from selenium. |  | The WebDriver should be imported. |
| **2.** | Open the Google Chrome browser. |  | The Google Chrome browser should be opened. |
| **3.** | **Maximize** the browser window. |  | The browser window should be maximized. |
| **4.** | Navigate to the **Google home page**. | https://www.google.com/ | The Google home page must be displayed. |
| **5.** | Identify the **Google search text box** and pass the value. | javatpoint | The value should be entered in the search text box. |
| **6.** | Click on the **Google search button**. |  | The Google search button should be clicked. |
| **7.** | Close the Browser. |  | The Browser should be closed. |

**Step1**

In the first step, we will type the following statement to import the web driver:

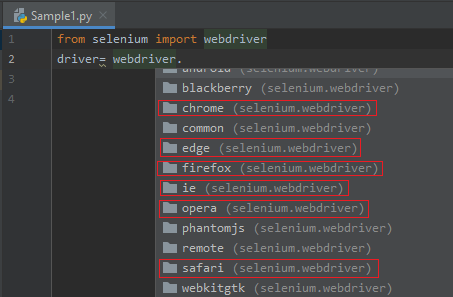
1. **from** selenium **import** webdriver



**Step2**

After that, we will open the Google Chrome browser.

As we can see in the below screenshot, we have multiple types of browsers options available, and we can select any browser from the list like **Chrome, Edge, firefox, Internet Explorer, opera, safari, etc**.



Following are the sample code for opening the Google Chrome browser:

1. driver = webdriver.Chrome()

**Step3**

In the next step, we will be maximizing our browser window size, and the sample code is as below:

1. driver.maximize\_window()

**Step4**

Then, we will navigate to the given URL.

The sample code is as below:

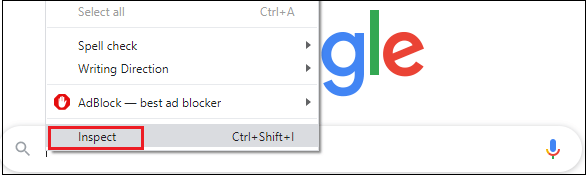
1. driver.get("https://www.google.com/")

#### **Note: As we know that Python is a very easy language to write code because we don't have to write multiple statements like as we did it java. Or if we want to comment out something, we just simply put a hash[#] in our statements, or we can directly press Ctrl+ Forward slash [/] from our keyboard.**

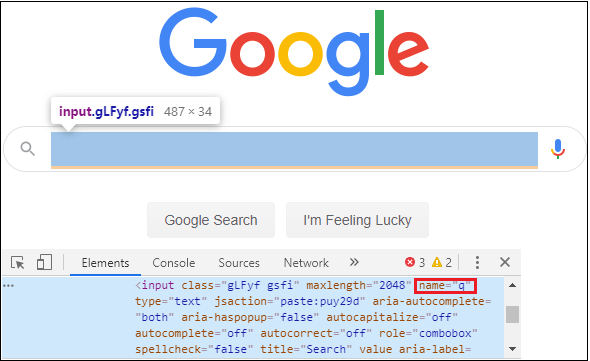
**Step5**

In this step, we are trying to locate the Google search text box with the help of its **Name** attribute value.

* Right-click on the **Google search** text box, and select the **Inspect** option in the pop-up menu as we can see in the below image:



* The developer tool window will be launched with all the specific codes used in the development of the **Google search** text box.
* And, copy the value of its **Name** attribute, that is "**q**" as we can see in the below image:



Here the sample code:

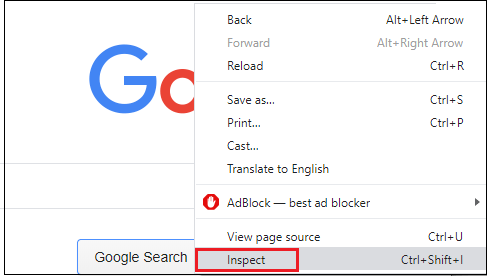
1. driver.find\_element\_by\_name("q").send\_keys("javatpoint")

**Step6**

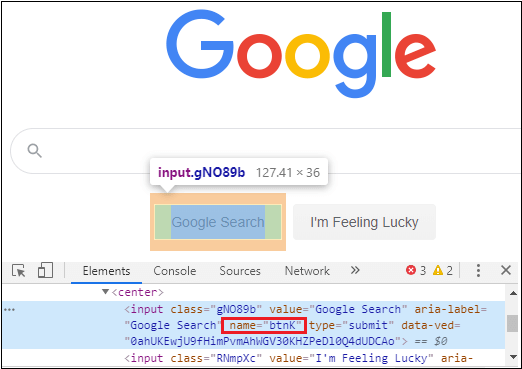
Once we identify the Google search text box, and we will identify the **Google Search button**.

So for this, follow the below process:

* Right-click on the **Google search**button, and select the **Inspect**option from the given pop-up menu as we can see in the below image:



* The developer tool window will be launched with having all the specific codes used in the development of the **Google search** button.
* Then, copy the value of its **name** attribute that is "**btnK**" as we can see in the below image:



And, the sample code is as following:

1. driver.find\_element\_by\_name("btnK").send\_keys(Keys.ENTER)

**Step7**

In the last step, we are closing the browser.

And, the sample code for closing the browser is as follows:

1. driver.close()

Our final test script will look like this, after completing all the above steps:

1. **from** Selenium **import** webdriver
2. **import** time
3. **from** Selenium.webdriver.common.keys **import** Keys
4. **print**("sample test case started")
5. driver = webdriver.Chrome()
6. #driver=webdriver.firefox()
7. #driver=webdriver.ie()
8. #maximize the window size
9. driver.maximize\_window()
10. #navigate to the url
11. driver.get("https://www.google.com/")
12. #identify the Google search text box and enter the value
13. driver.find\_element\_by\_name("q").send\_keys("javatpoint")
14. time.sleep(3)
15. #click on the Google search button
16. driver.find\_element\_by\_name("btnK").send\_keys(Keys.ENTER)
17. time.sleep(3)
18. #close the browser
19. driver.close()
20. **print**("sample test case successfully completed")

#### **Note: Import time: Time is a Python module, which is used to handle the time-related tasks such as time.sleep().**

**from Selenium.webdriver.common.keys import Keys:**

Here, we are adding Keys libraries from Selenium, like in the above code, we are using the **Enter** key instead of **click()** method to perform a particular scenario.

### **Run and validate the test scripts**

Once we are done with writing the Selenium test script, we will run our test scripts.

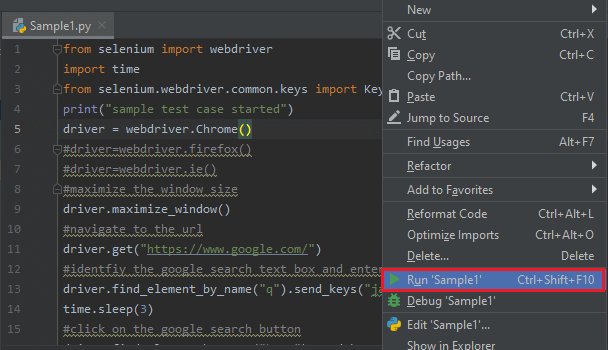
Here we will run our test scripts in two ways:

* **Run in Python IDE**
* **Run in Command Prompt**

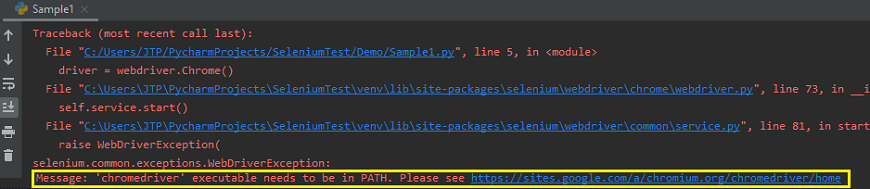
**Run in Python IDE**

So, for this first, we will see how to run the Selenium test script in Python IDE.

* Right-click on the code, and select **Run 'Sample1'** from the popup menu as we can see in the below screenshot:

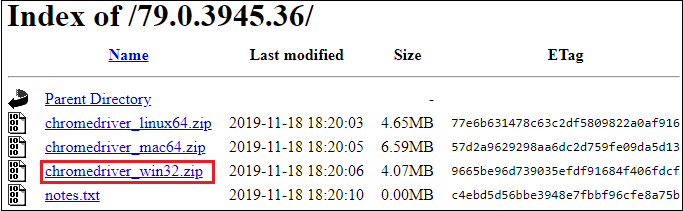


* When we run this script it will give an exception because we don't have the Chrome driver executable file as we can in the below image:



To overcome this exception, we will download the chrome driver executable from below link: <https://chromedriver.storage.googleapis.com/index.html?path=79.0.3945.36/>

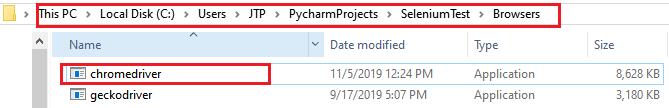
* Once we click on the above link, we will click on the**zip file**based upon our operating system platform. Like we have**Windows platform**that's why we clicked on the**zip**to download the Executable file as we can see in the below screenshot:



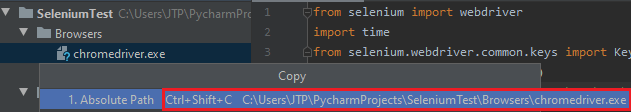
* After downloading the **exe** file, we can paste this file to the Python folder and unzip it.
* Then, we will create one more folder called libraries as **Browsers** in the Python IDE.
* Right-click on the Project**(SeleniumTest) → New → Directory** as we can see in the below screenshot:

Selenium with Python Tutorial

* And, we will add all the driver's executable files in the **Browsers** folder manually.
* For this, we will copy the **chrome driver exe** file from the **Python folder**, and paste in the **Browser** folder as we can see in the below image:



* Now go to **PyCharm** IDE, and copy the **Absolute path** of chromdriver.exe file as we can see in the below screenshot:

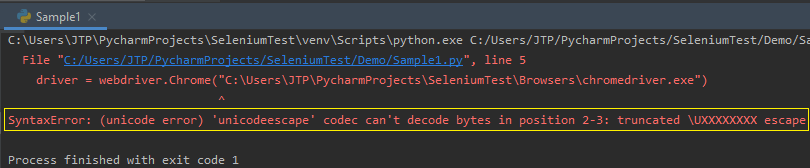


* Replace the statement "driver = webdriver.Chrome()" with a statement given below:

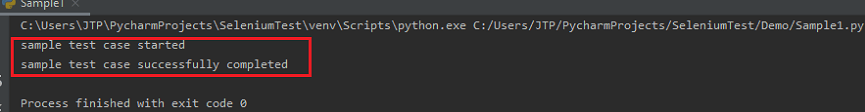
1. driver=webdriver.Chrome(r"C:\Users\JTP\PycharmProjects\SeleniumTest\Browsers\chromedriver.exe")

#### **Note: Here, we will use "r" to overcome the Unicode error.**

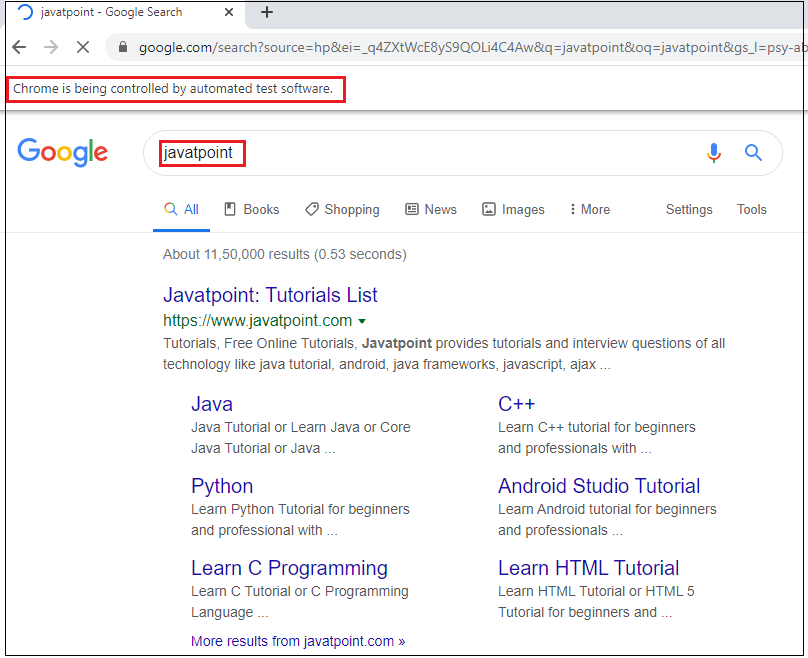
As we can see in the below screenshot, if we do not put r in the code, it will generate the **Syntax Error**.



* After that, we will run the **sample1** once again, and it will execute the code successfully as we can see in the below image:



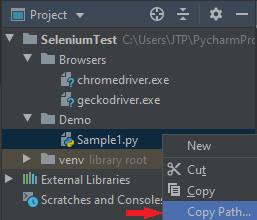
The above test script will launch the Google Chrome browser and automate all the test scenarios.



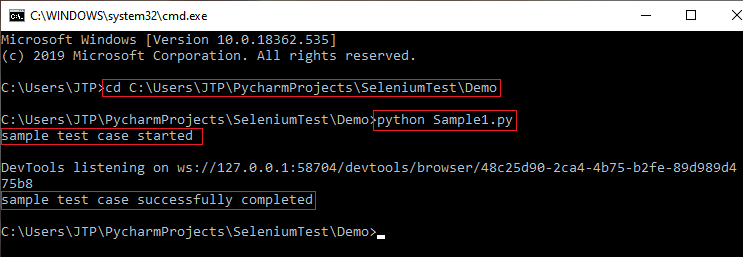
**Run in Command Prompt**

To run the above test script in the Command prompt, follow the below process:

* Copy the location of the **Sample1.py** file as we can see in the below image:



* And paste in the command Prompt, first go to the particular folder then enter the below command:  
  **Python Sample1.py**
* Then, press the **Enter** key as we can see in the below screenshot that the **sample test case stared.**
* And after automating all the scenarios, it will show the message as a **sample test case successfully completed**.



**Programs**

**Code-1: For opening website**

from selenium import webdriver  
from selenium.webdriver.common.keys import Keys

#Name : Bhavesh Kewalramani  
#Roll No.: A-25

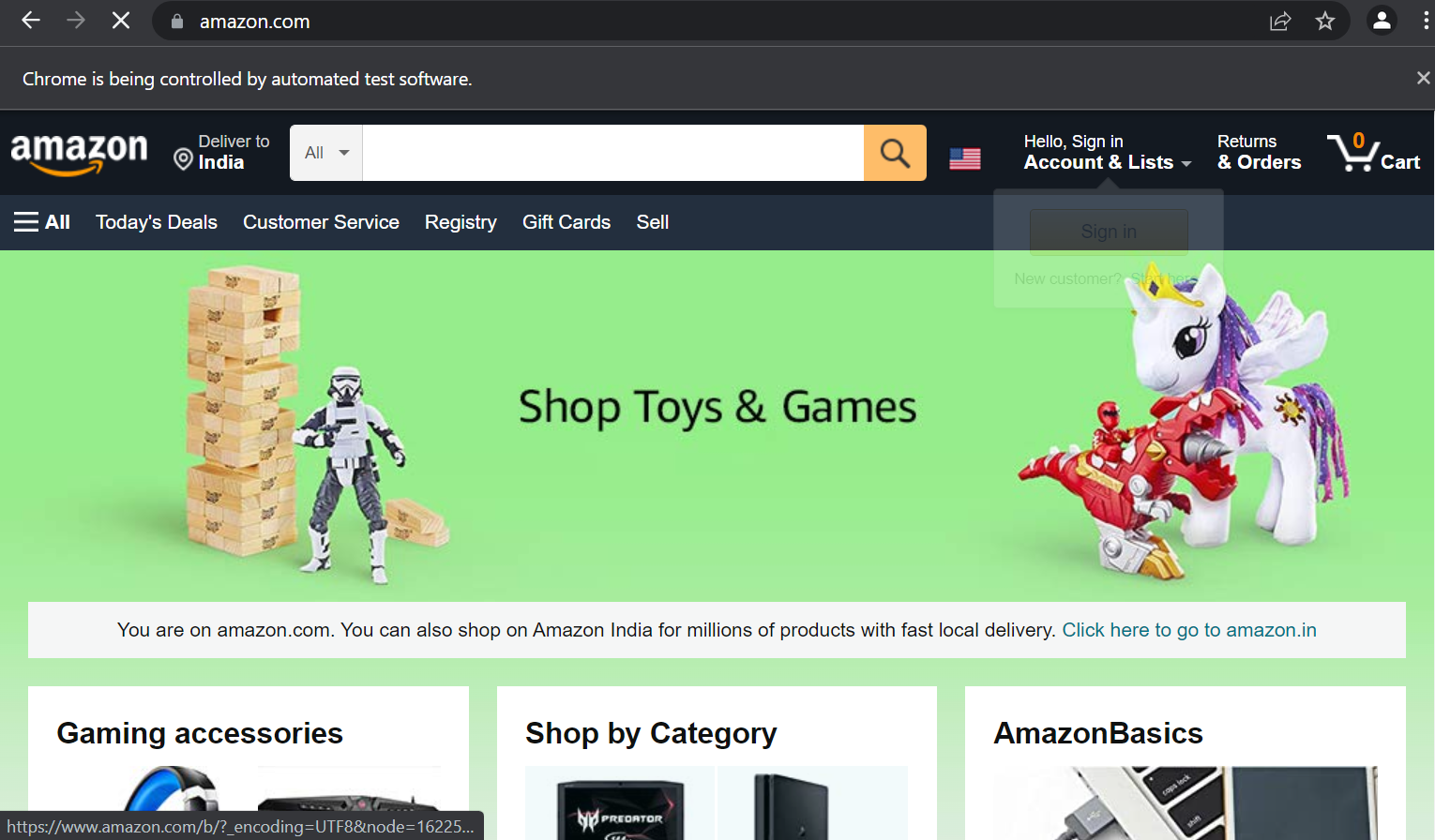
#Section : A

#Semester: VI

#Shift : I

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Chrome \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Chrome ... ")  
driverChrome = webdriver.Chrome(executable\_path=r"C:\Drivers\chromedriver\_win32\chromedriver.exe")  
driverChrome.get("https://www.python.org")  
print(driverChrome.title)  
driverChrome.get("https://www.amazon.com/")  
print(driverChrome.title)  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Mozilla \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Firefox ... ")  
driverFirefox = webdriver.Firefox(executable\_path=r"C:\Drivers\geckodriver-v0.30.0-win64\geckodriver.exe")  
driverFirefox.get("https://www.python.org")  
print(driverFirefox.title)  
driverFirefox.get("https://www.amazon.com/")  
print(driverFirefox.title)

**Output:**

****

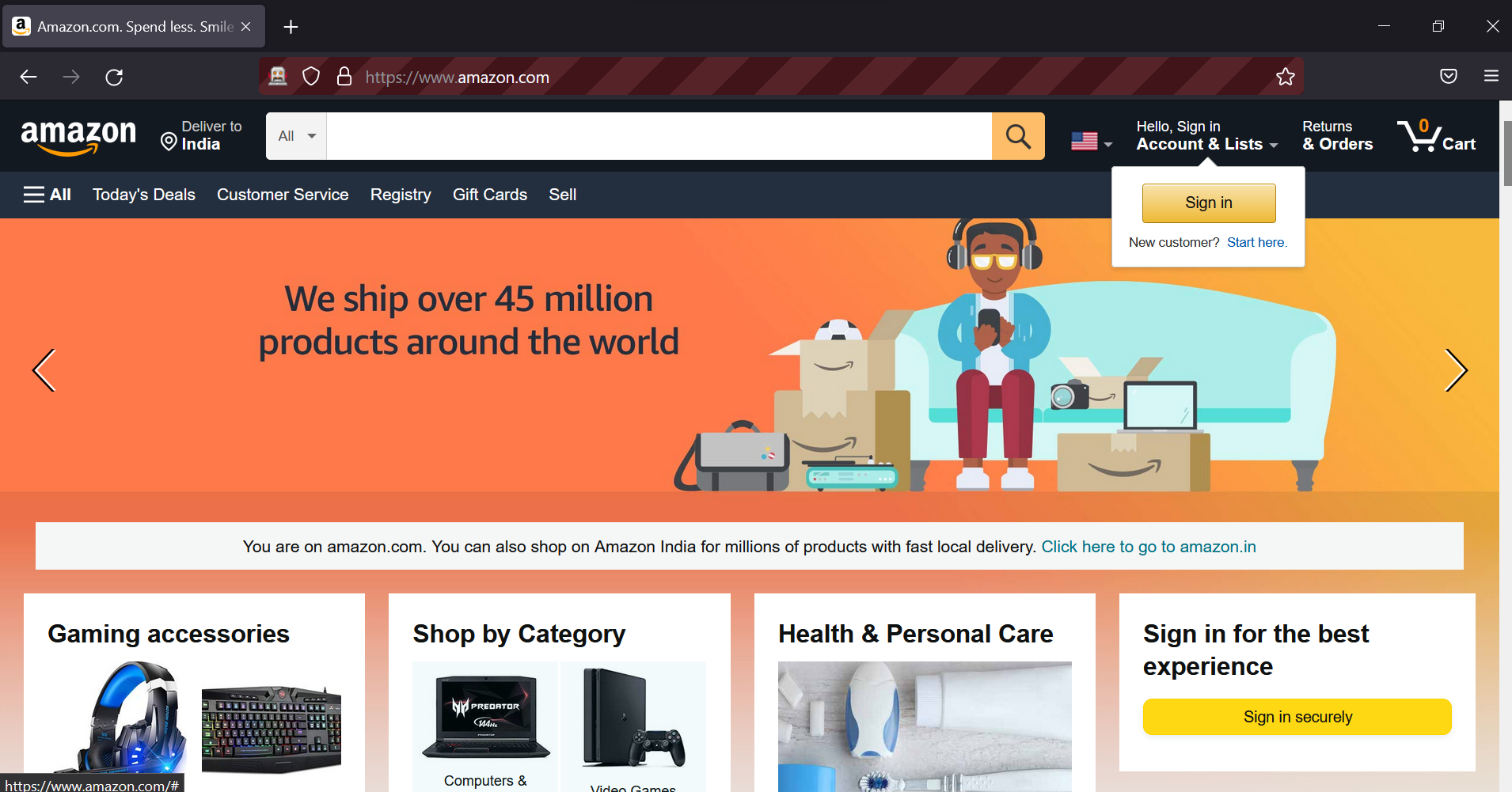
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

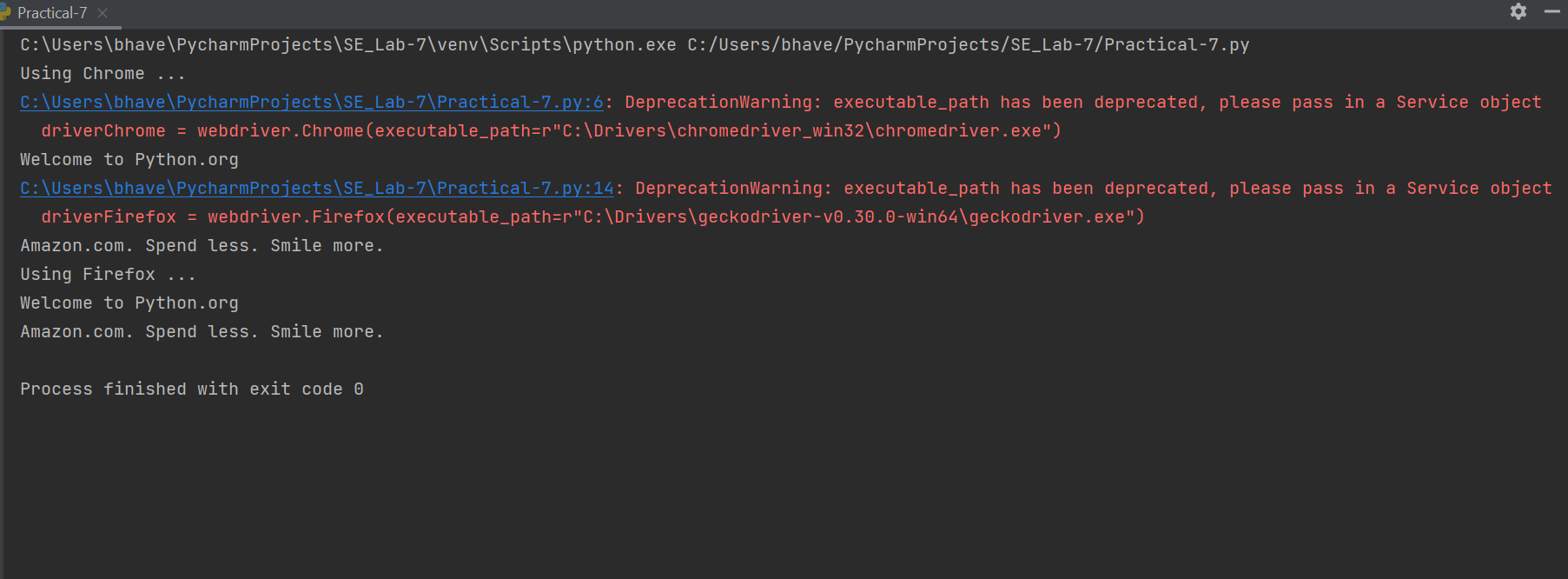
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

**Code-2: Detecting and giving input to text boxes**

from selenium import webdriver  
from selenium.webdriver.common.keys import Keys  
import time

#Name : Bhavesh Kewalramani  
#Roll No.: A-25

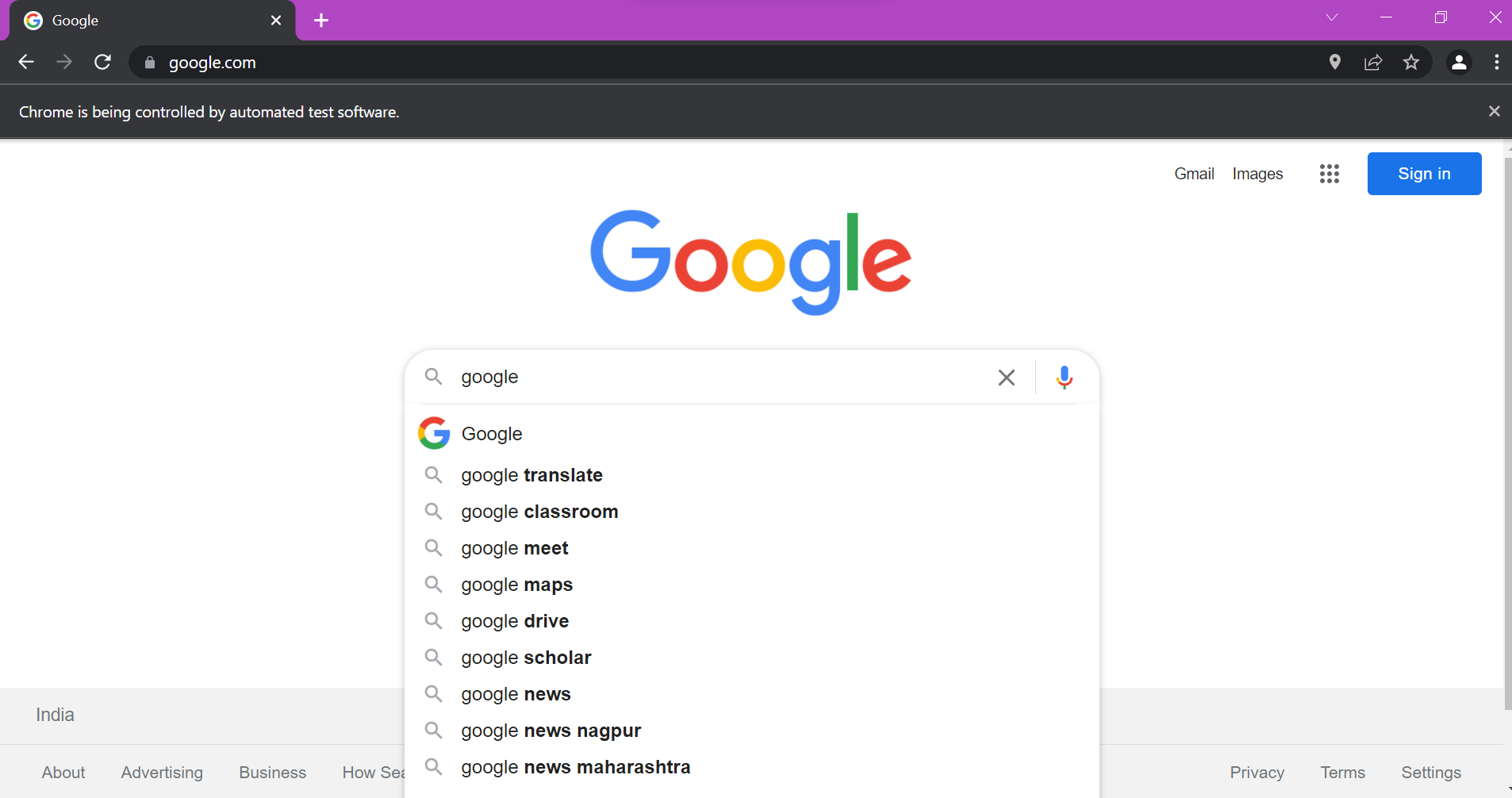
#Section : A

#Semester: VI

#Shift : I

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Chrome \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Chrome ... ")  
driverChrome = webdriver.Chrome(executable\_path=r"C:\Drivers\chromedriver\_win32\chromedriver.exe")  
driverChrome.implicitly\_wait(0.5)  
driverChrome.get("https://www.google.com/")  
  
l= driverChrome.find\_element\_by\_name("q")  
l.send\_keys("google")  
  
print("Value of input box: " + l.get\_attribute('value'))  
time.sleep(5)  
l.send\_keys(Keys.ENTER)  
  
time.sleep(5)  
driverChrome.close()  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Mozilla \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Firefox ... ")  
driverFirefox = webdriver.Firefox(executable\_path=r"C:\Drivers\geckodriver-v0.30.0-win64\geckodriver.exe")  
driverFirefox.implicitly\_wait(0.5)  
driverFirefox.get("https://www.google.com/")  
  
l= driverFirefox.find\_element\_by\_name("q")  
l.send\_keys("google")  
  
print("Value of input box: " + l.get\_attribute('value'))  
time.sleep(5)  
l.send\_keys(Keys.ENTER)

**Output:**

****

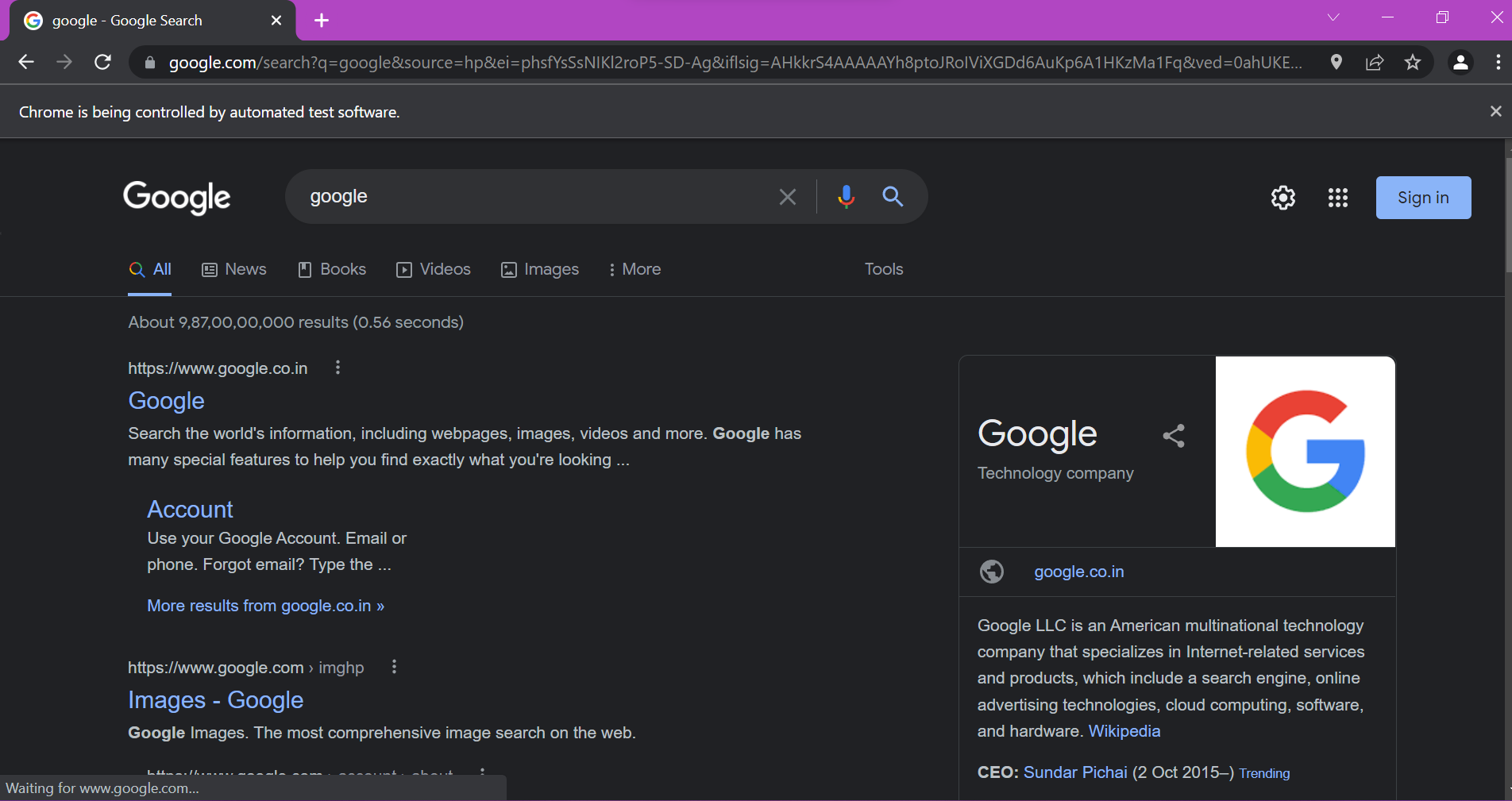
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

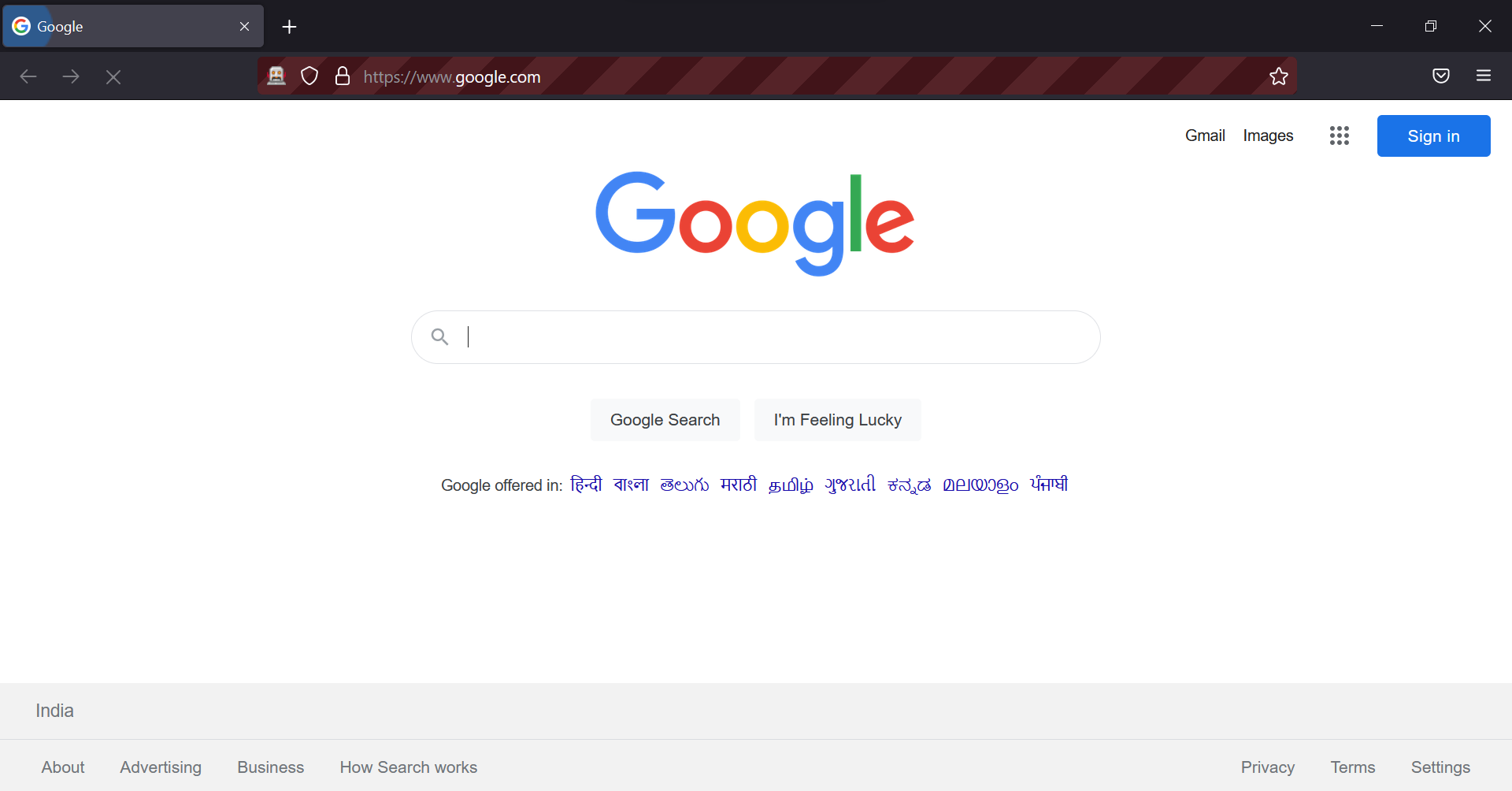
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

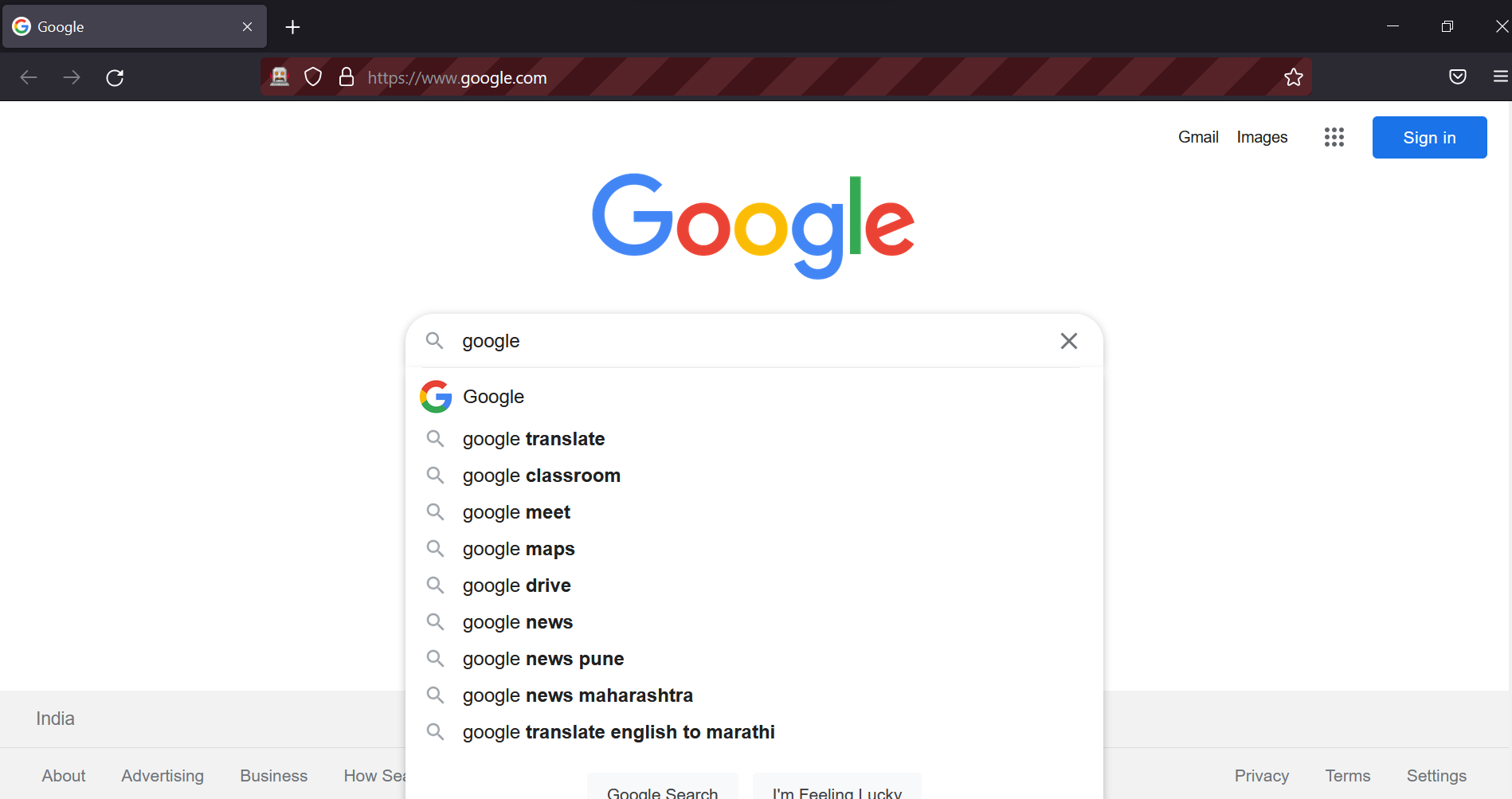
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

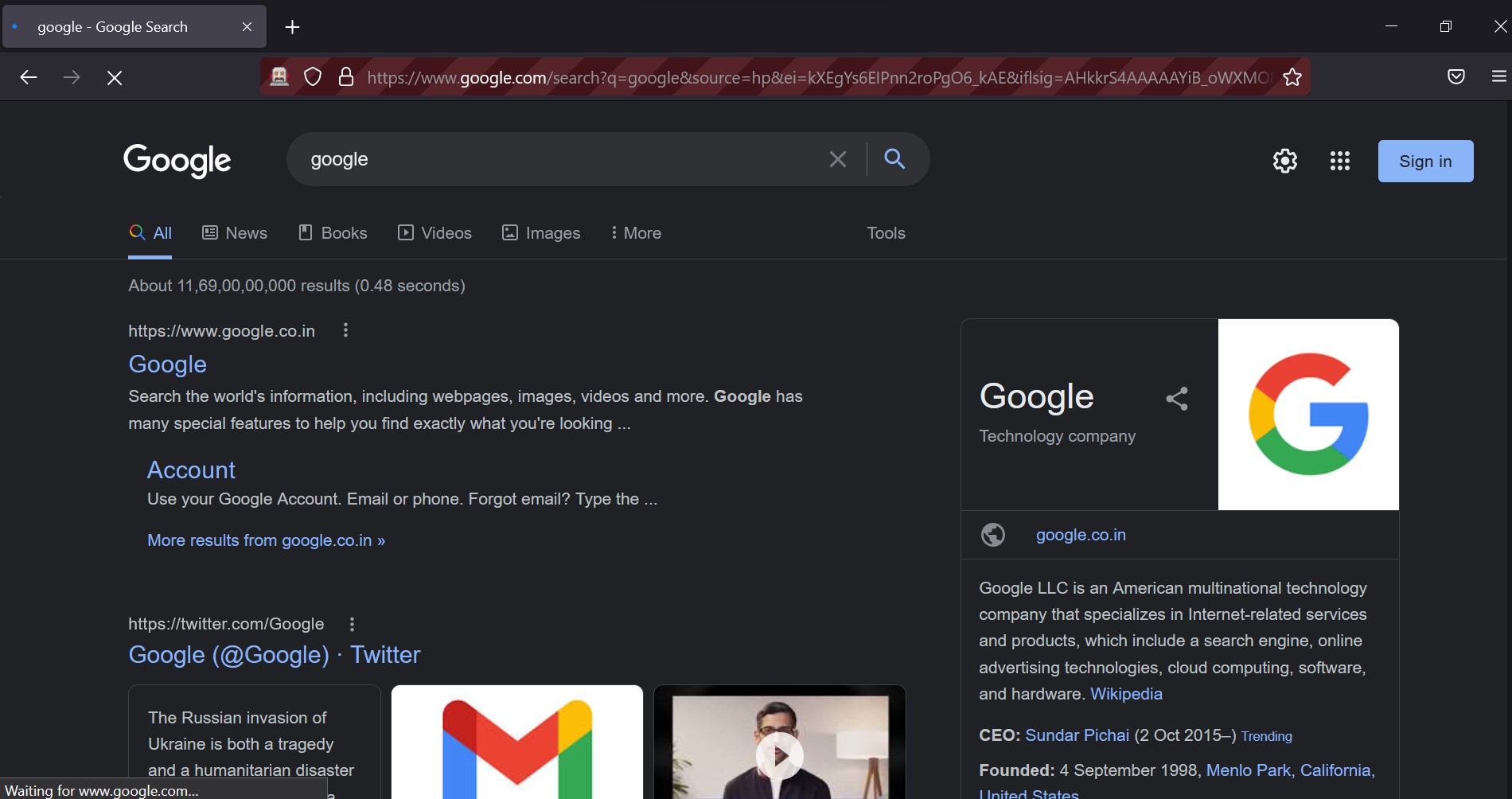
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

**Code-3: Testing Radio buttons and Checkboxes**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.keys import Keys  
import time

#Name : Bhavesh Kewalramani  
#Roll No.: A-25

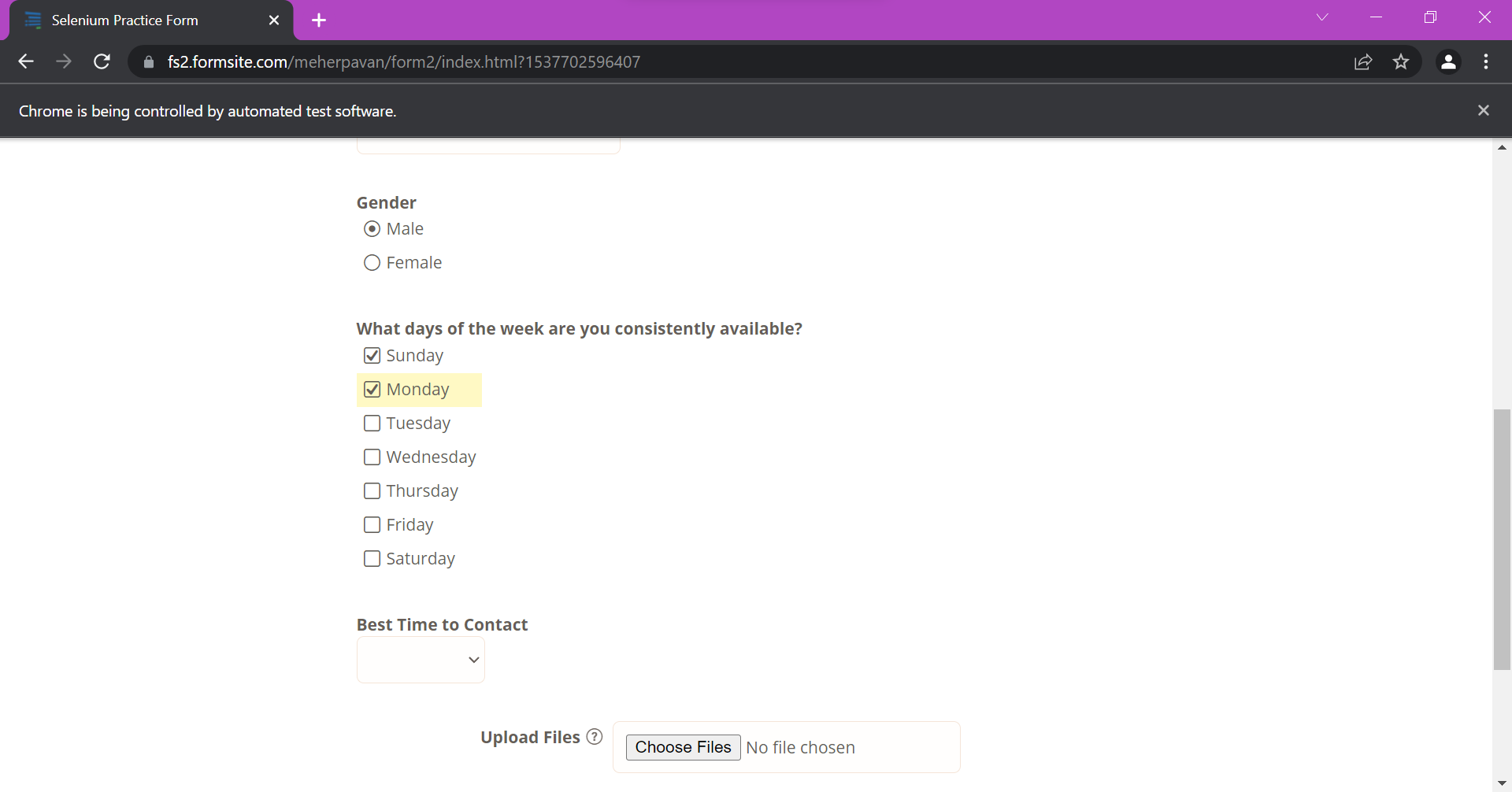
#Section : A

#Semester: VI

#Shift : I

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Chrome \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Chrome ... ")  
driverChrome = webdriver.Chrome(executable\_path=r"C:\Drivers\chromedriver\_win32\chromedriver.exe")  
driverChrome.get("https://fs2.formsite.com/meherpavan/form2/index.html?1537702596407")  
driverChrome.find\_element\_by\_xpath('//\*[@id="q26"]/table/tbody/tr[1]/td/label').click()  
driverChrome.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[1]/td/label').click()  
driverChrome.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[2]/td/label').click()  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Mozilla \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Firefox ... ")  
driverFirefox = webdriver.Firefox(executable\_path=r"C:\Drivers\geckodriver-v0.30.0-win64\geckodriver.exe")  
driverFirefox.get("https://fs2.formsite.com/meherpavan/form2/index.html?1537702596407")  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q26"]/table/tbody/tr[1]/td/label').click()  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[1]/td/label').click()  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[2]/td/label').click()

**Output:**

****

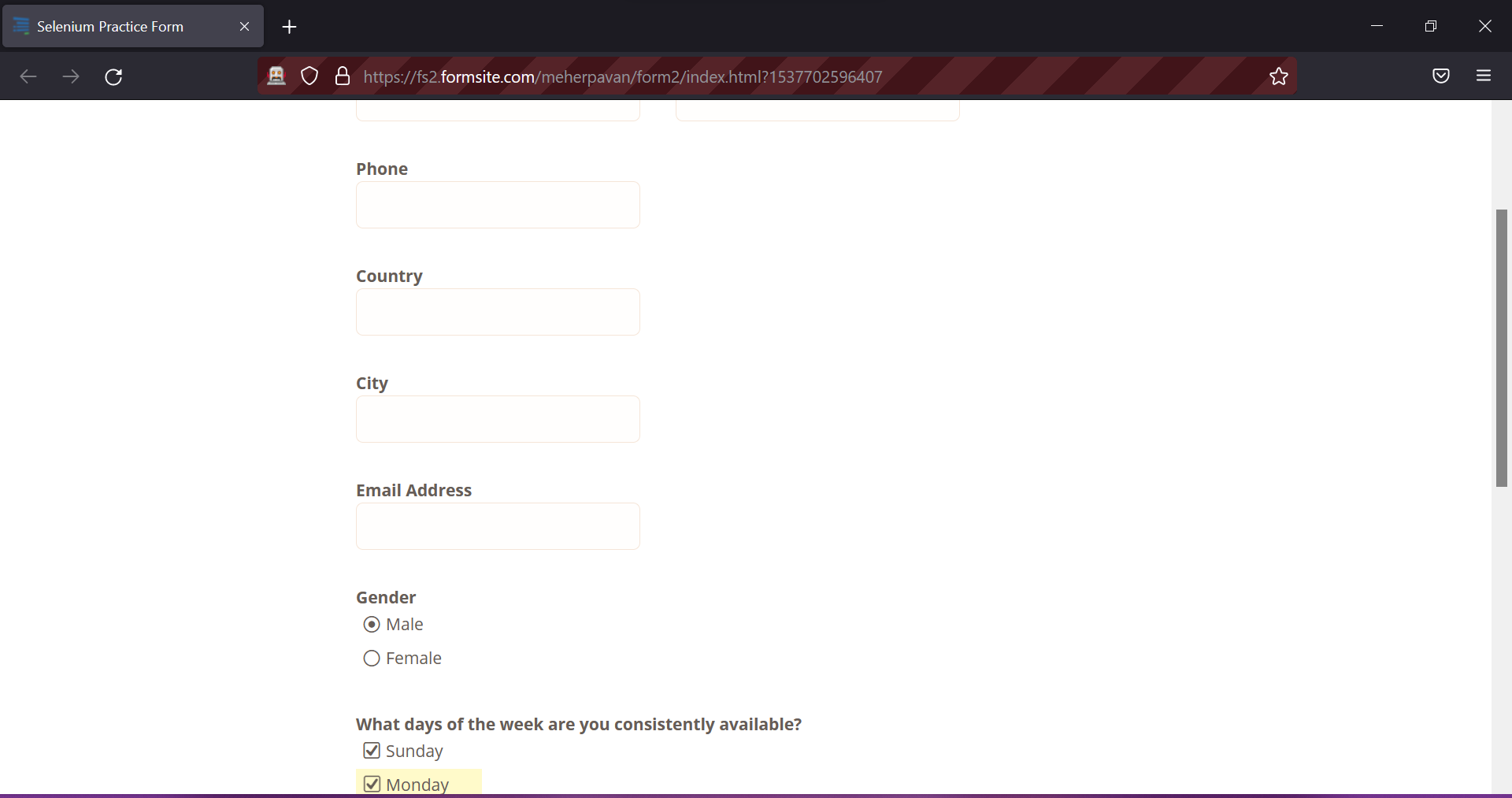
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



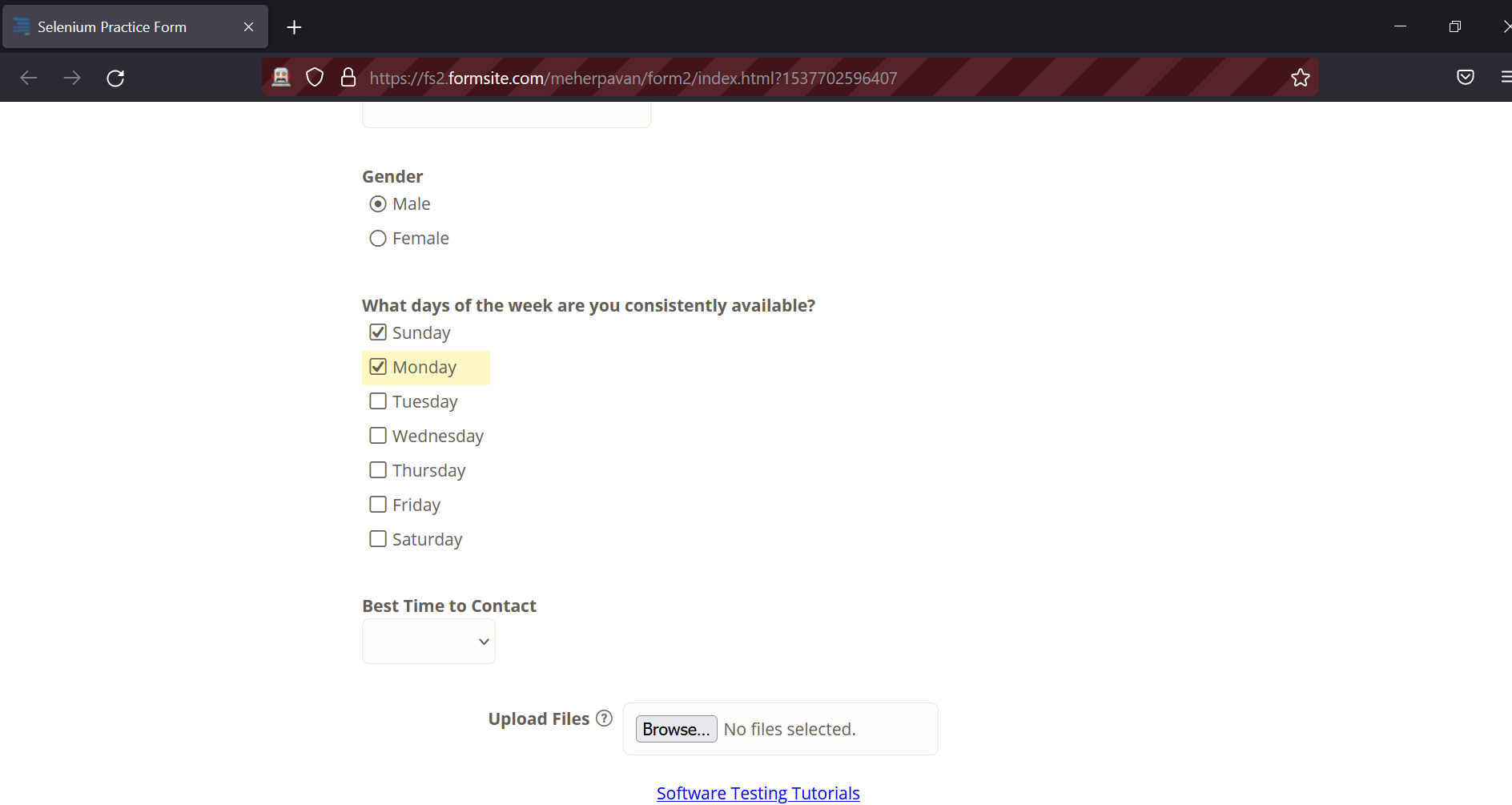
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



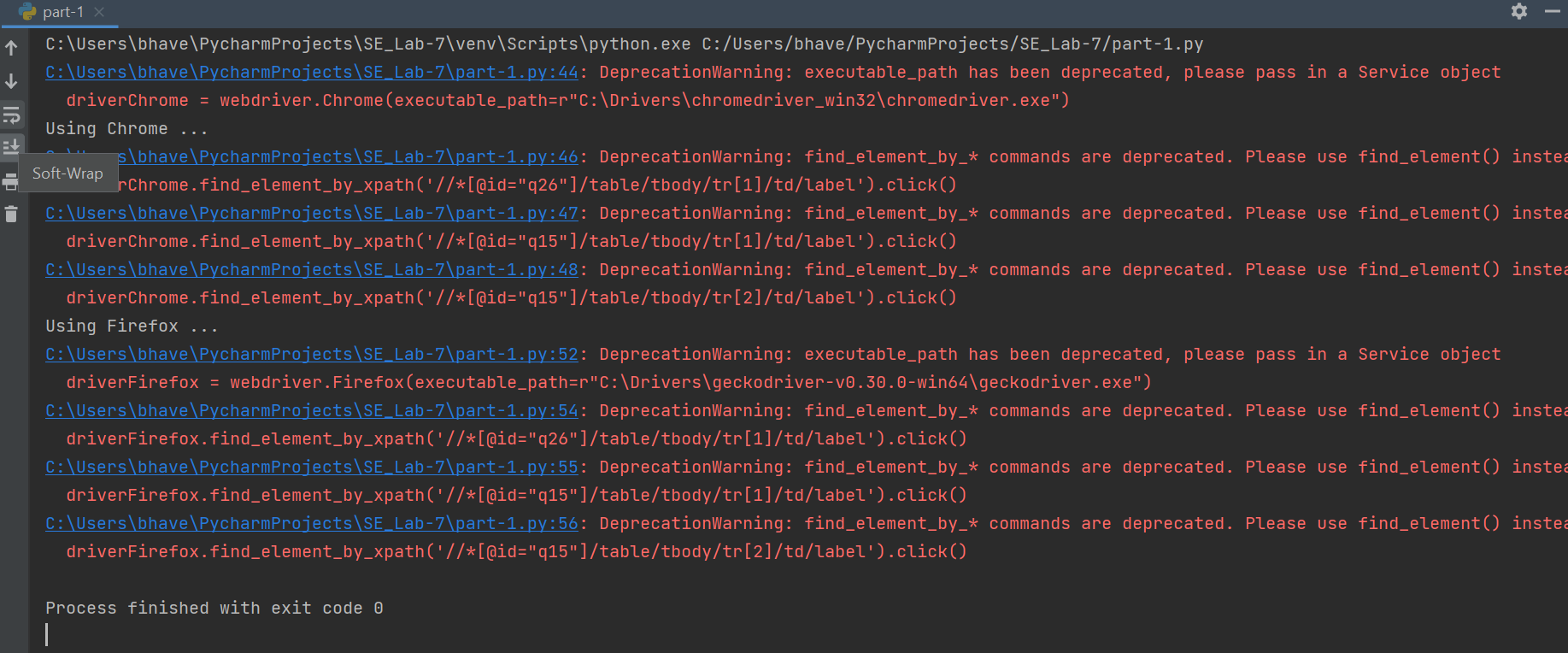
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

**Code-4: Giving input to Login form**

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  
import time

#Name : Bhavesh Kewalramani  
#Roll No.: A-25

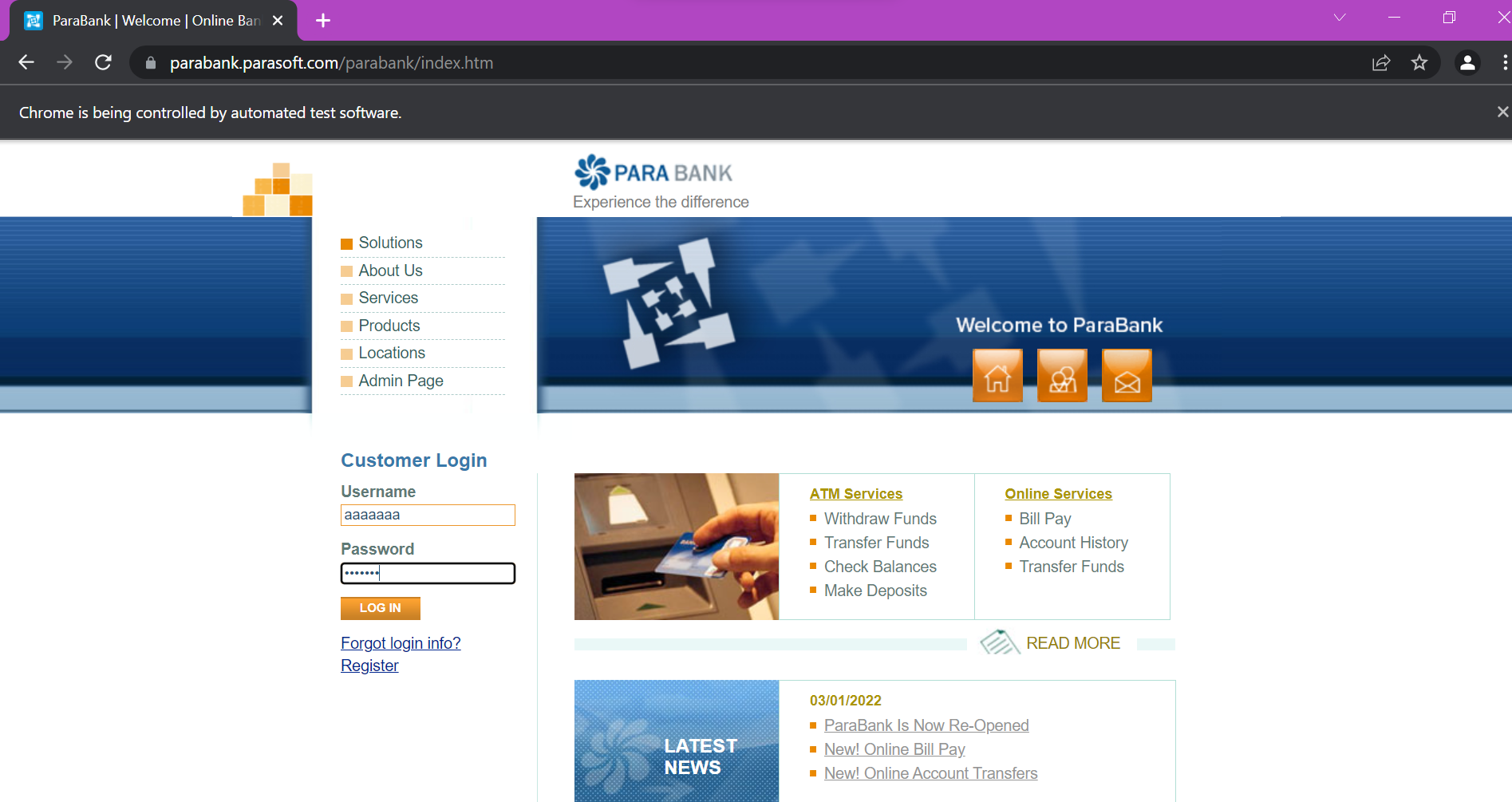
#Section : A

#Semester: VI

#Shift : I

usernameStr = 'aaaaaaa'  
passwordStr = 'aaaaaaa'  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Chrome \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Chrome ... ")  
driverChrome = webdriver.Chrome(executable\_path=r"C:\Drivers\chromedriver\_win32\chromedriver.exe")  
driverChrome.get("https://parabank.parasoft.com/parabank/index.htm")  
  
username = driverChrome.find\_element\_by\_name('username')  
username.send\_keys(usernameStr)  
password = driverChrome.find\_element\_by\_name('password')  
password.send\_keys(passwordStr)  
time.sleep(20)  
driverChrome.find\_element\_by\_class\_name('button').click()  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Mozilla \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Firefox ... ")  
driverFirefox = webdriver.Firefox(executable\_path=r"C:\Drivers\geckodriver-v0.30.0-win64\geckodriver.exe")  
driverFirefox.get("https://parabank.parasoft.com/parabank/index.htm")  
  
username = driverFirefox.find\_element\_by\_name('username')  
username.send\_keys(usernameStr)  
password = driverFirefox.find\_element\_by\_name('password')  
password.send\_keys(passwordStr)  
time.sleep(20)  
driverFirefox.find\_element\_by\_class\_name('button').click()

**Output:**

****

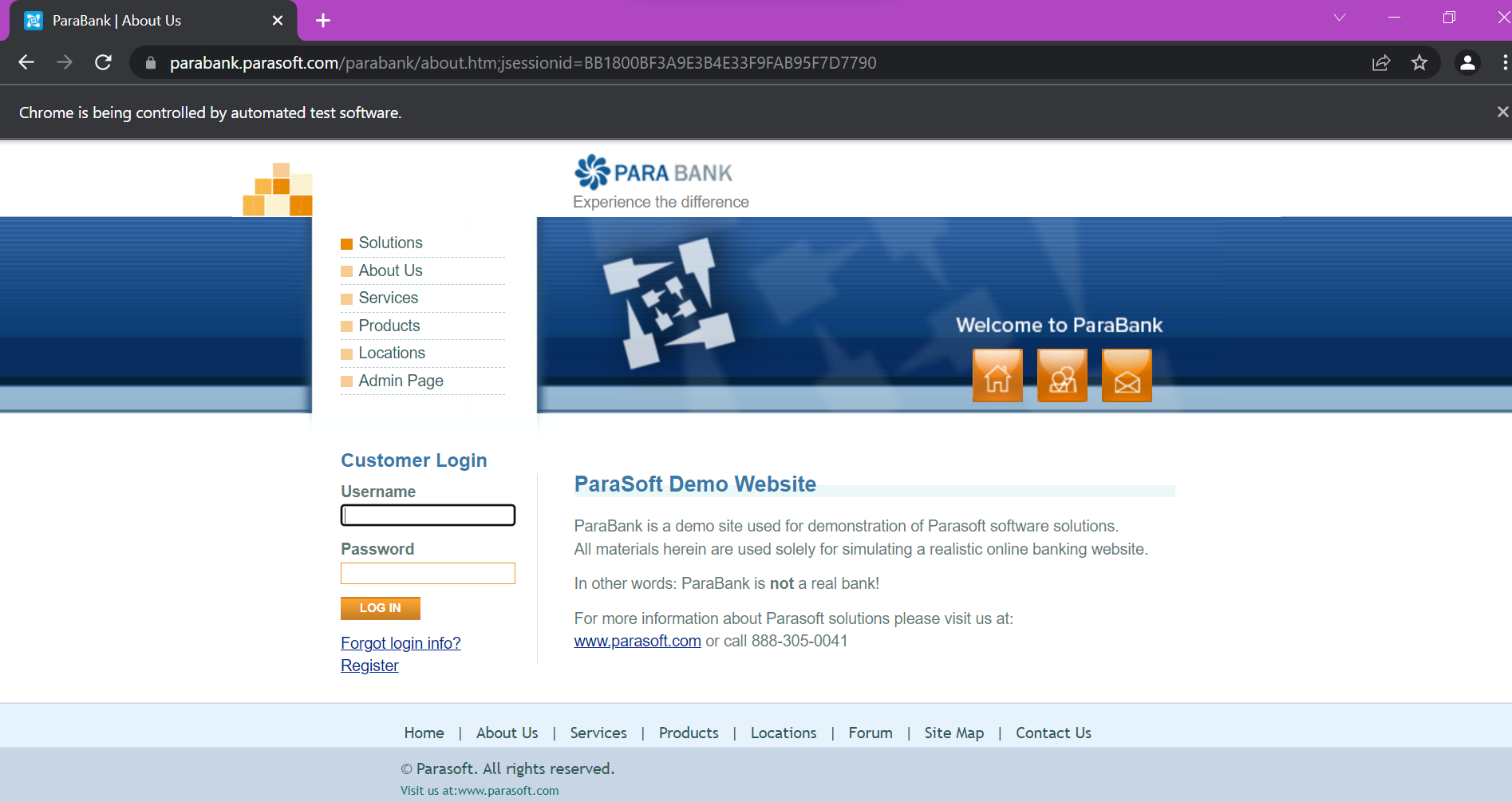
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

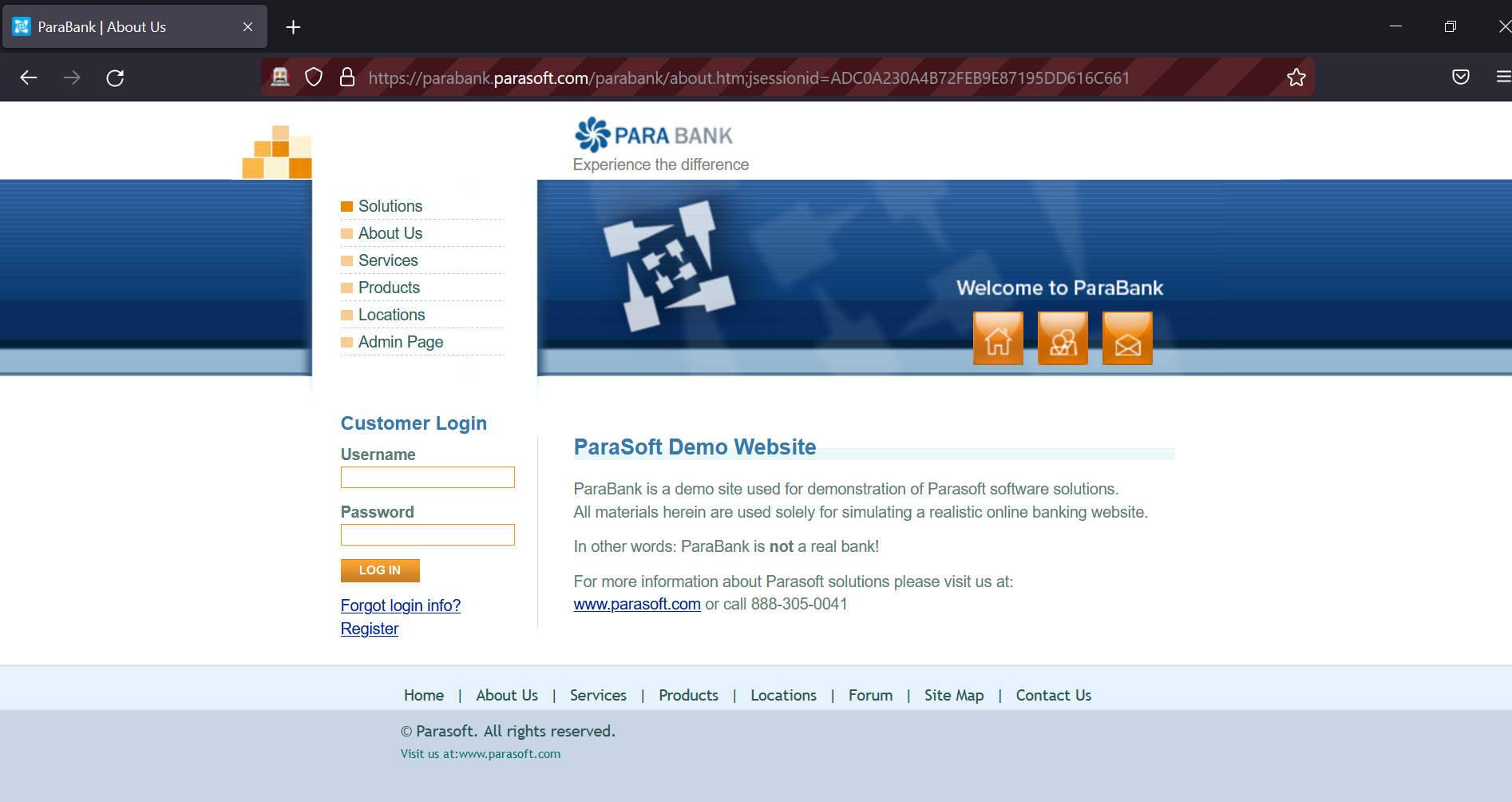
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

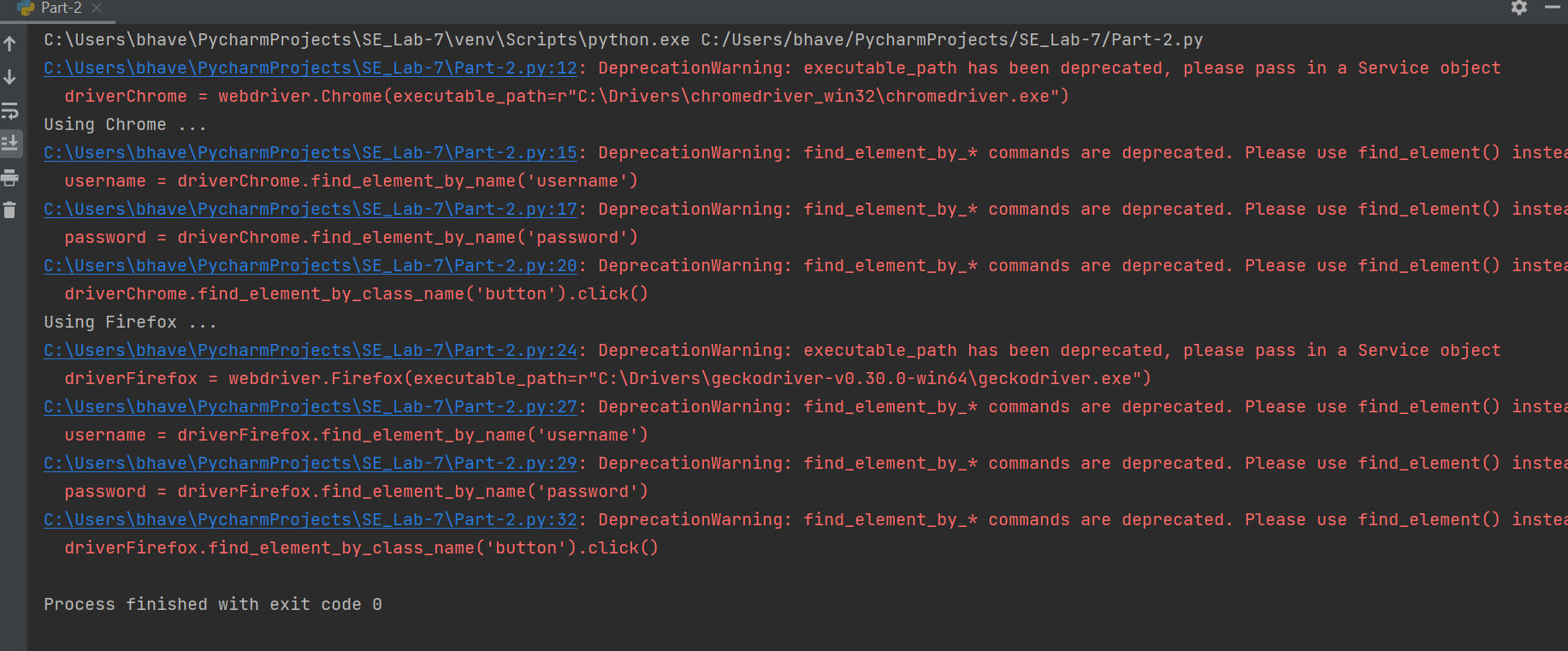
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

**Code-5: Filling form automatically**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.common.keys import Keys  
from selenium.webdriver.support.ui import Select  
import time

#Name : Bhavesh Kewalramani  
#Roll No.: A-25

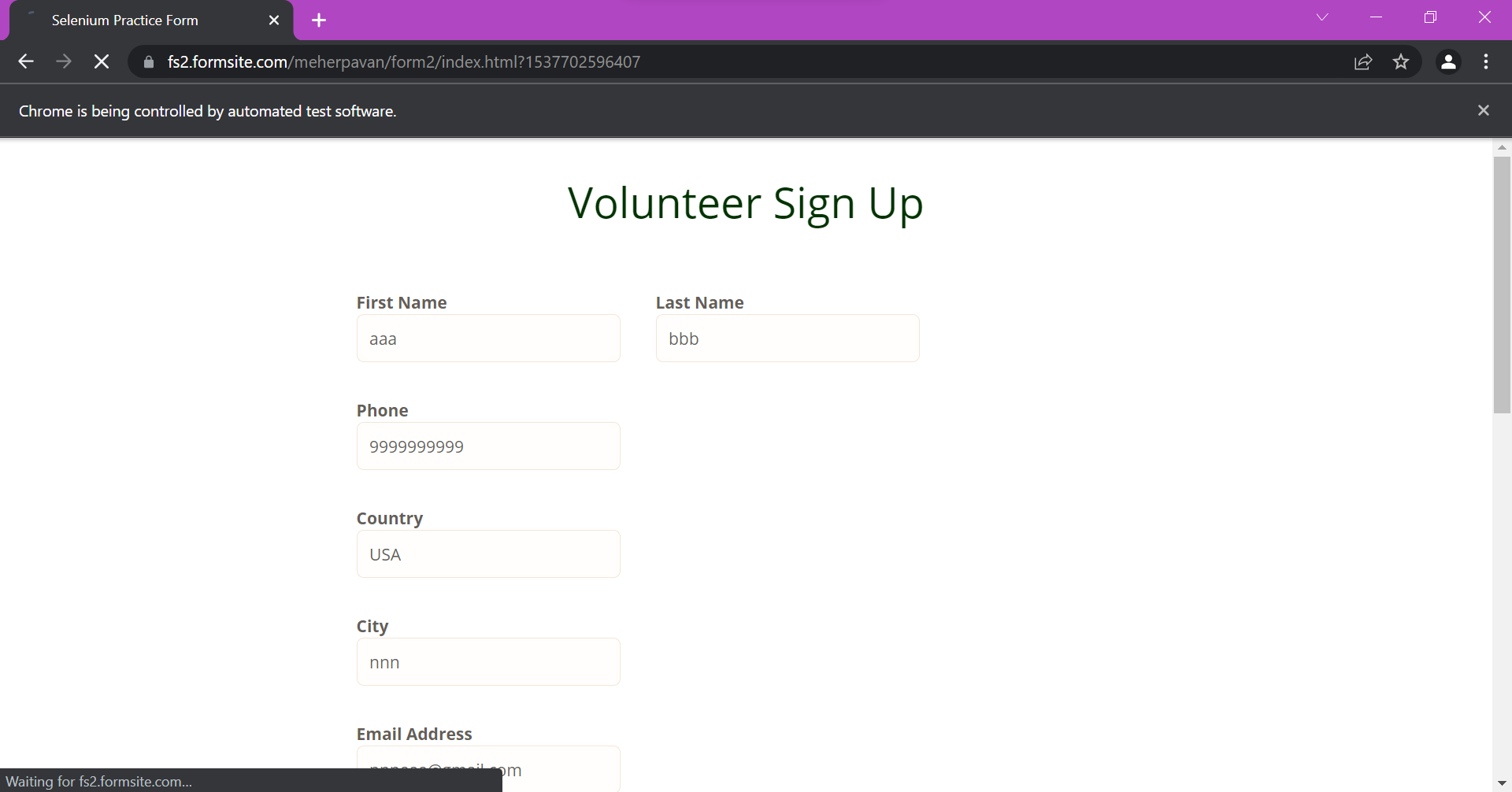
#Section : A

#Semester: VI

#Shift : I

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Chrome \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Chrome ... ")  
driverChrome = webdriver.Chrome(executable\_path=r"C:\Drivers\chromedriver\_win32\chromedriver.exe")  
driverChrome.get("https://fs2.formsite.com/meherpavan/form2/index.html?1537702596407")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-1")  
l.send\_keys("aaa")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-2")  
l.send\_keys("bbb")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-3")  
l.send\_keys("9999999999")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-4")  
l.send\_keys("USA")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-5")  
l.send\_keys("nnn")  
l = driverChrome.find\_element\_by\_name("RESULT\_TextField-6")  
l.send\_keys("nnnaaa@gmail.com")  
driverChrome.find\_element\_by\_xpath('//\*[@id="q26"]/table/tbody/tr[1]/td/label').click()  
driverChrome.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[1]/td/label').click()  
driverChrome.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[2]/td/label').click()  
select = Select(driverChrome.find\_element\_by\_id('RESULT\_RadioButton-9'))  
select.select\_by\_visible\_text('Evening')  
s = driverChrome.find\_element\_by\_xpath("//input[@type='file']")  
s.send\_keys("C:\\Users\\bhave\\Downloads\\sample.jpg")  
Select(driverChrome.find\_element\_by\_id('FSsubmit')).click()  
  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* For Firefox \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
print("Using Firefox ... ")  
driverFirefox = webdriver.Firefox(executable\_path=r"C:\Drivers\geckodriver-v0.30.0-win64\geckodriver.exe")  
driverFirefox.get("https://fs2.formsite.com/meherpavan/form2/index.html?1537702596407")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-1")  
l.send\_keys("aaa")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-2")  
l.send\_keys("bbb")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-3")  
l.send\_keys("9999999999")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-4")  
l.send\_keys("USA")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-5")  
l.send\_keys("nnn")  
l = driverFirefox.find\_element\_by\_name("RESULT\_TextField-6")  
l.send\_keys("nnnaaa@gmail.com")  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q26"]/table/tbody/tr[1]/td/label').click()  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[1]/td/label').click()  
driverFirefox.find\_element\_by\_xpath('//\*[@id="q15"]/table/tbody/tr[2]/td/label').click()  
select = Select(driverFirefox.find\_element\_by\_id('RESULT\_RadioButton-9'))  
select.select\_by\_visible\_text('Evening')  
s = driverFirefox.find\_element\_by\_xpath("//input[@type='file']")  
s.send\_keys("C:\\Users\\bhave\\Downloads\\sample.jpg")  
Select(driverFirefox.find\_element\_by\_id('FSsubmit')).click()

**Output:**

****

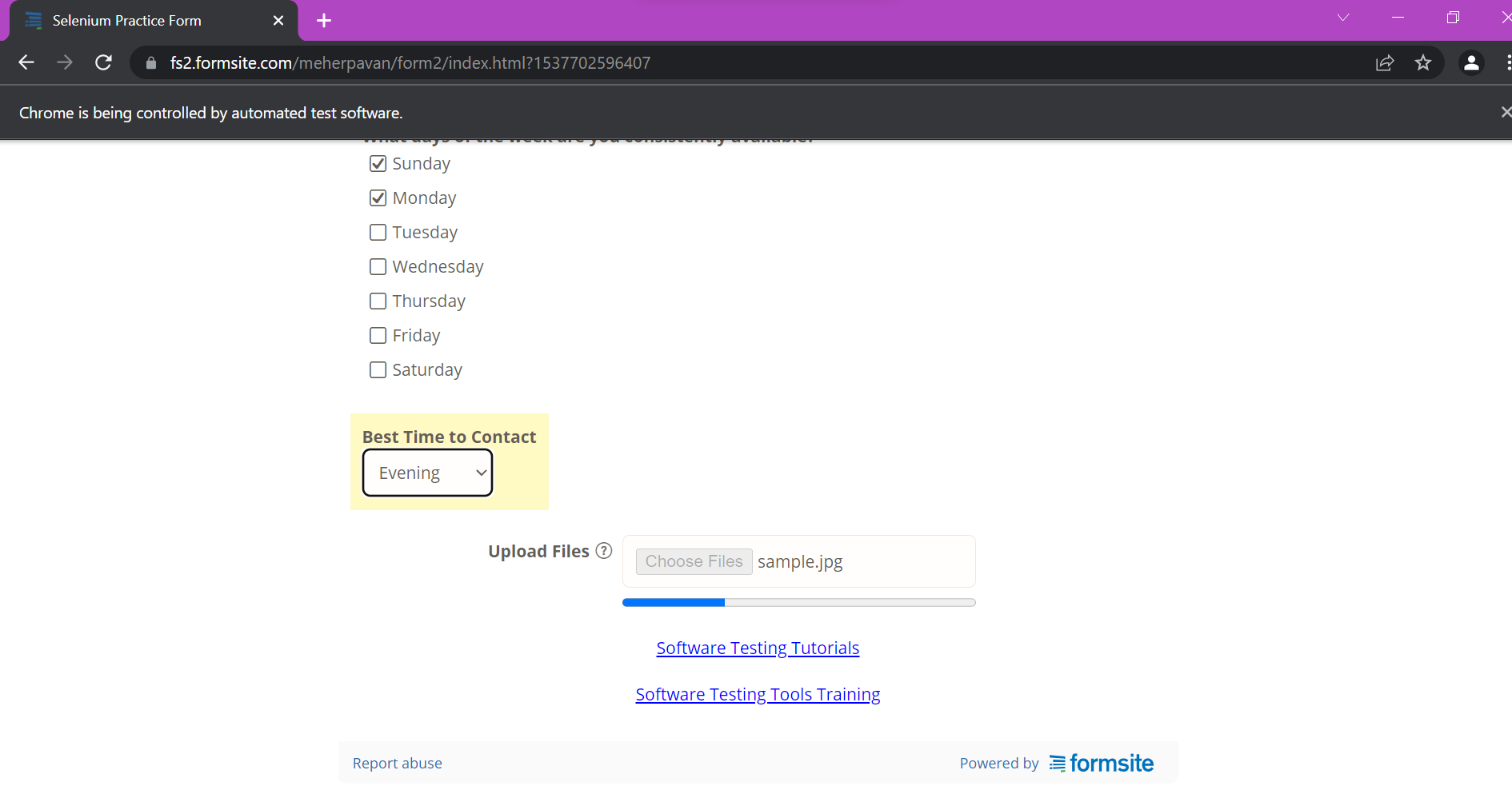
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

****

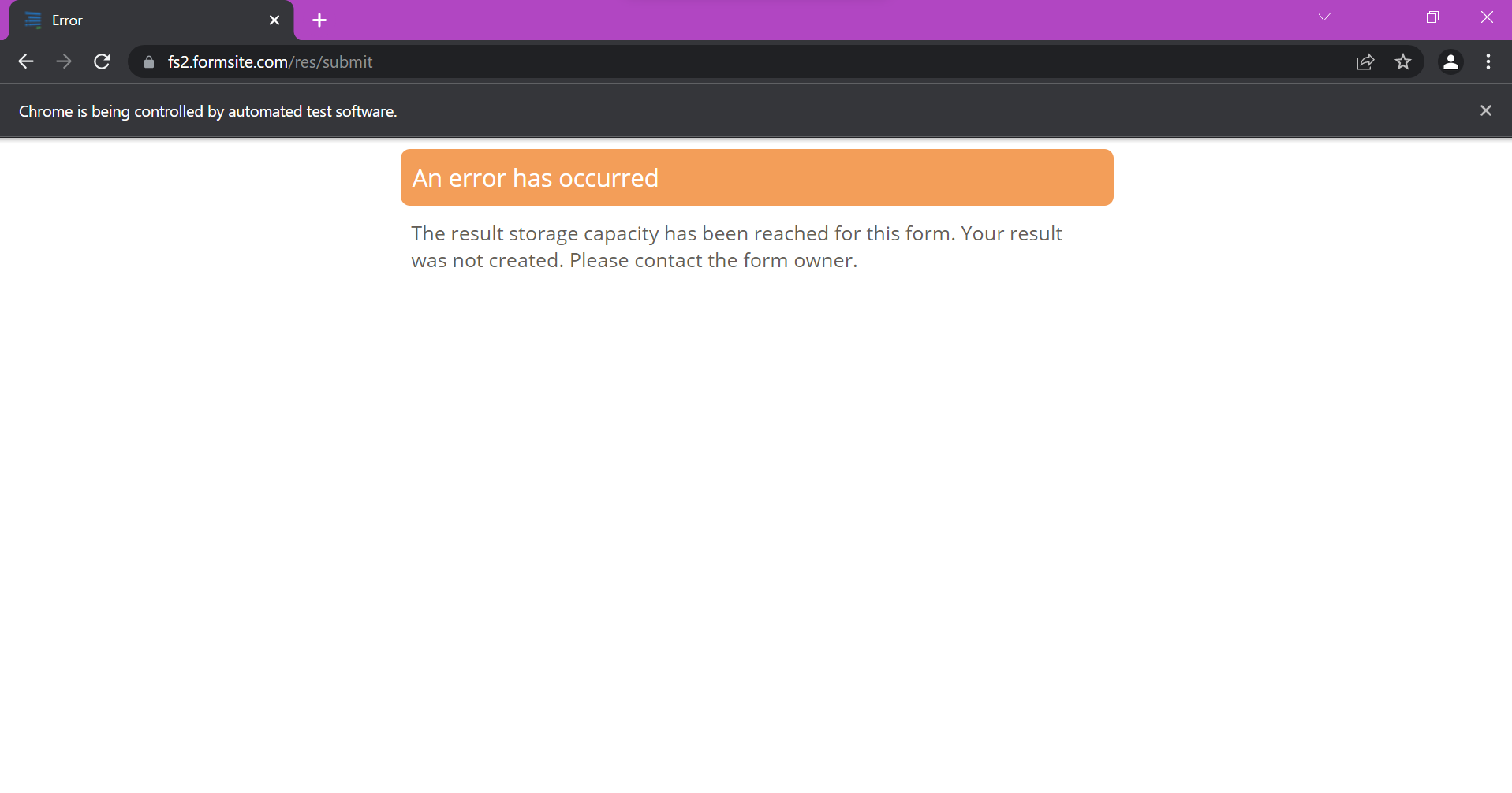
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



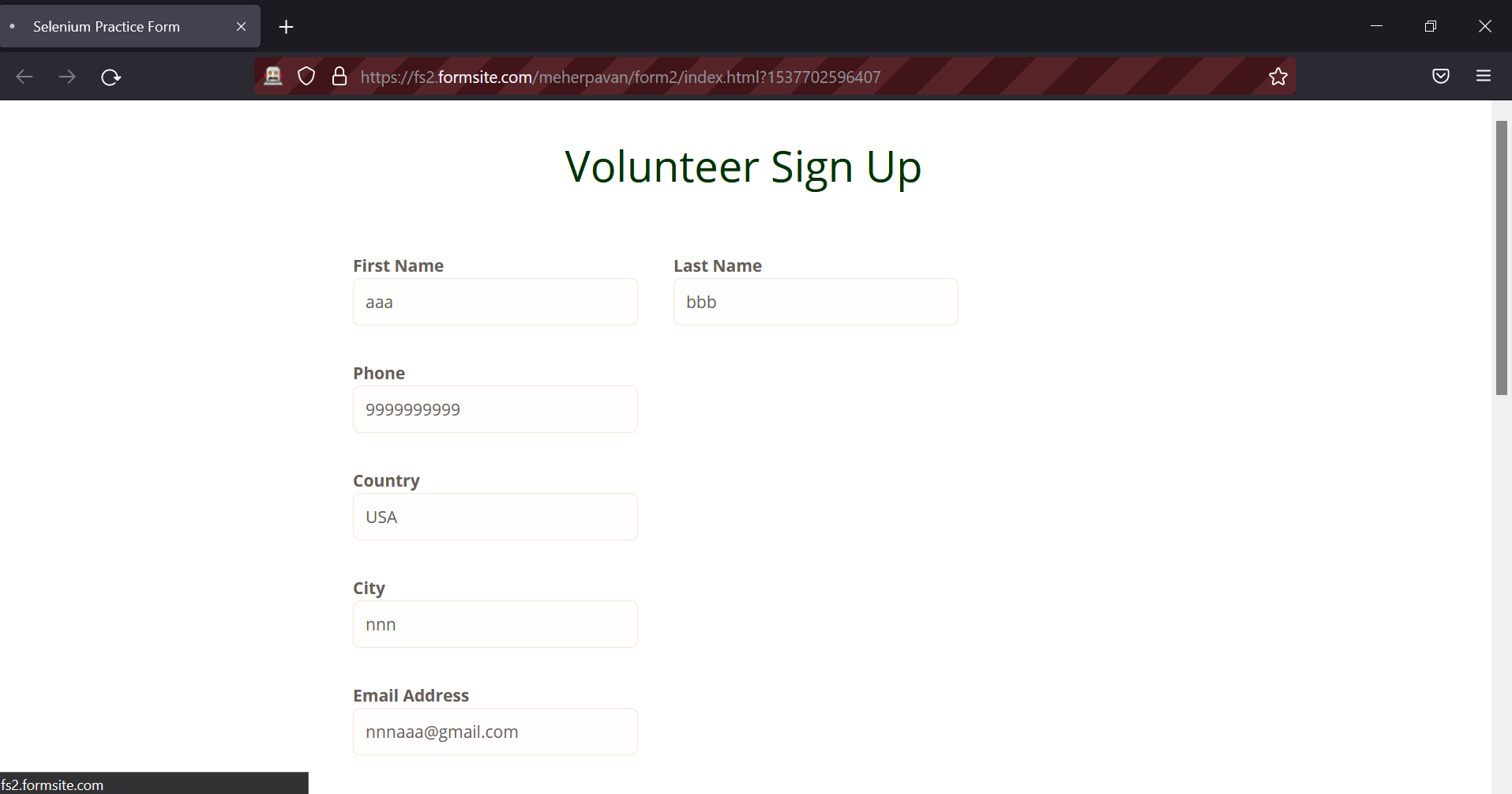
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



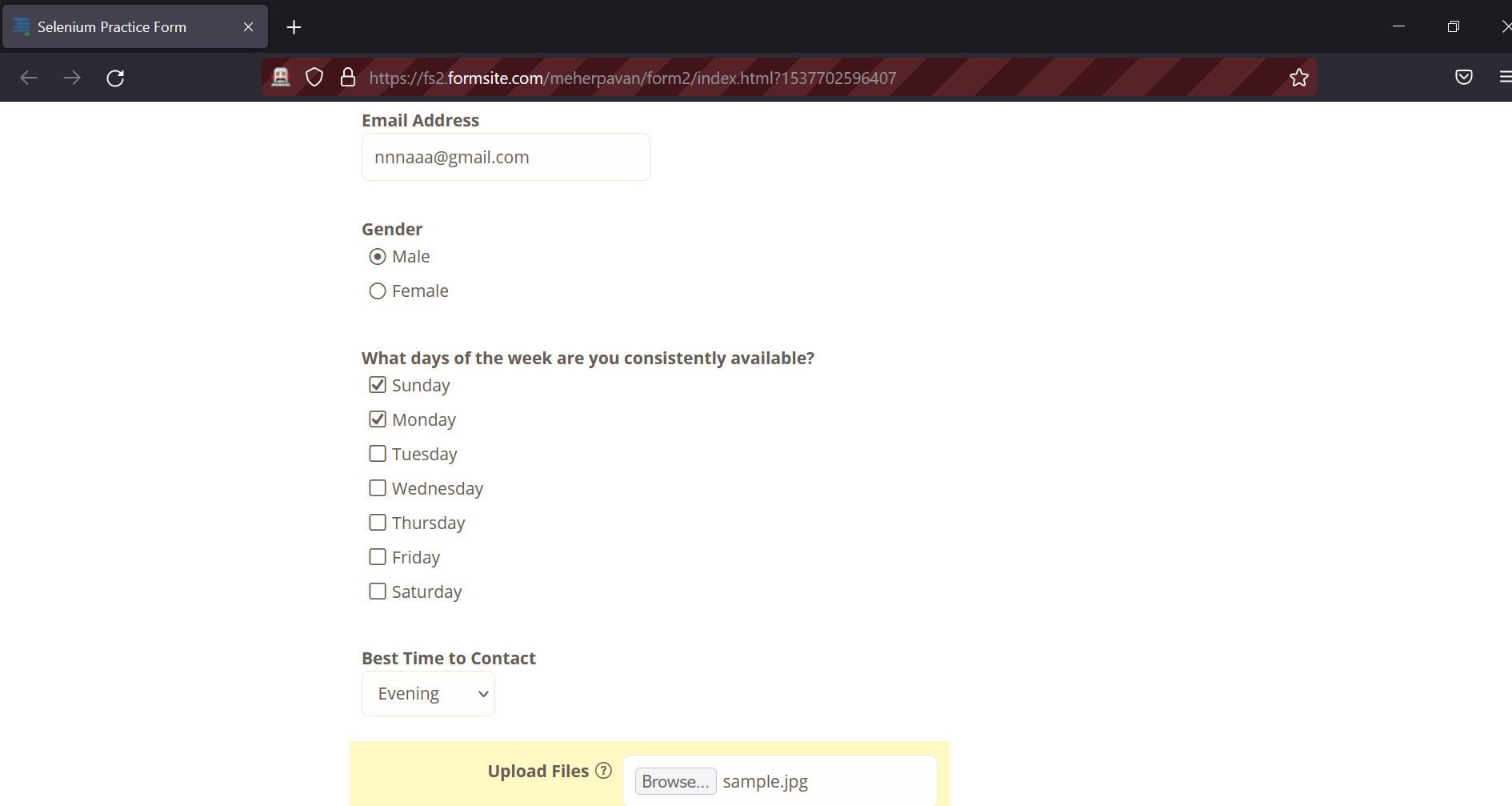
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



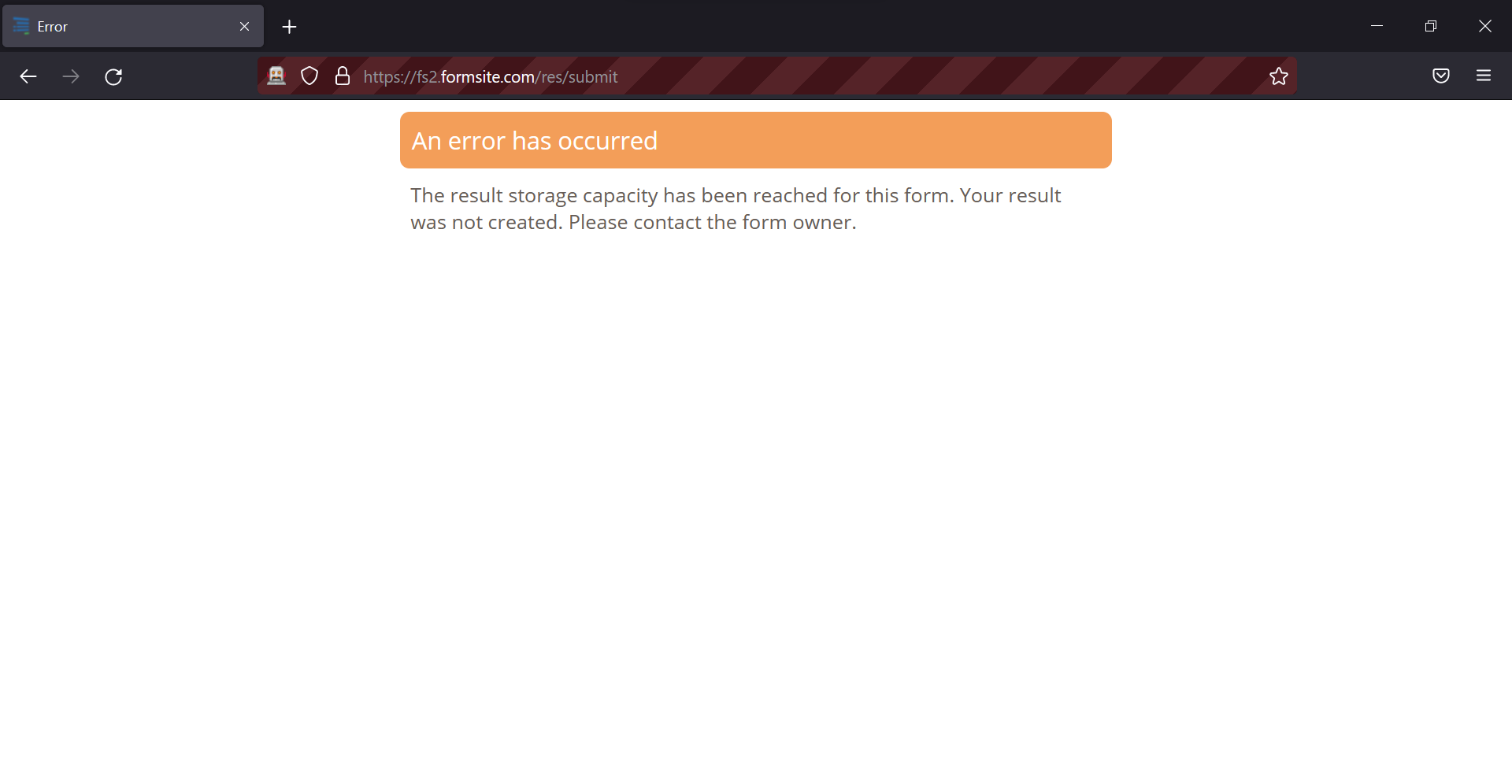
Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I



Name: Bhavesh Kewalramani

Roll No.: A-25

Section: A

Semester: VI

Shift: I

**Conclusion:**

Selenium is a cost-effective and flexible tool developers can use in the automation testing of their web applications.

The most intriguing feature of this software is the ability to test applications across various web browsers. This ensures that websites do not crash or breakdown in certain browsers.

The Selenium software is ideal for companies developing applications that support heavy traffic, especially social platforms and e-commerce websites. This software has undergone improvement over the years, which has added value to web development.