Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Experiment 5a

Selenium
(Automated Testing using Selenium)

Name : Parth Savla **Roll No: D135**

Aim: To Set Up Selenium and Perform Basic Web UI Testing

Theory:

Introduction to Selenium:

Selenium is an open-source automation framework for web applications, enabling testers and developers to automate browser interactions and perform functional testing. With versatile tools like WebDriver, Selenium supports various programming languages and facilitates cross-browser testing, making it a go-to choice for efficient and scalable web automation.

Why Use Selenium?

- **Free and Open Source**
- **Cross-browser support** (Chrome, Firefox, Edge, etc.)
- **Supports multiple languages** (Java, Python, C#, JavaScript)
- **Automates repetitive browser tasks** like:
    - Clicking buttons
    - Filling forms
    - Navigating pages
- **Integrates with tools** like Jenkins, GitHub Actions for CI/CD
- Supports automation of **functional testing, regression testing, and data-driven testing**.

Different Types of Selenium Tests

- **Functional Tests** − These tests help to check various new functionalities and features of the application or the product under test.
- **Regression Tests** − These tests help to check if code changes have broken an existing functionality of the application.
- **Smoke Tests** − These tests help to verify if a new build is stable enough to carry out testing activities on it.
- **Integration Tests** − These tests help to validate if the integration of all the modules are working together as a unit.
- **Unit Tests** − These tests are created by the developers to test their code.

Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Core Components of Selenium

- **Selenium WebDriver**
  - Core component.
  - Directly communicates with the browser to perform actions.
- **Selenium IDE**
  - A simple browser extension (Chrome/Firefox).
  - Records and plays back user interactions for quick tests.
- **Selenium Grid**
  - Runs tests on multiple machines and browsers in parallel.
  - Saves time in large-scale testing.

I. Install Selenium on a Linux VM running in Google Cloud Platform (GCP).

1. Create & Connect to a VM
   1. Go to Google Cloud Console.
   2. Create a Compute Engine VM Instance:
      a. Choose Ubuntu / Debian / CentOS as the OS.
      b. Select a machine type (e.g., e2-micro for light use).
   3. Connect via SSH from the Console or using your terminal:

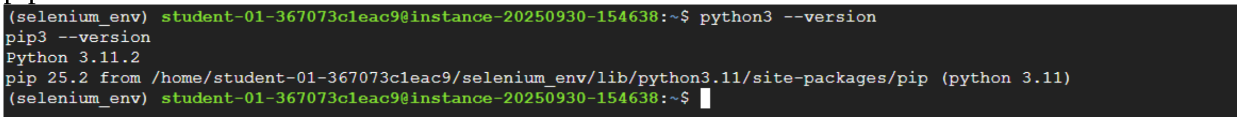
      gcloud compute ssh <INSTANCE_NAME>

2. Update Packages: sudo apt update && sudo apt upgrade -y

3. Install Python & Pip
(If not already installed)

sudo apt install python3 python3-pip -y

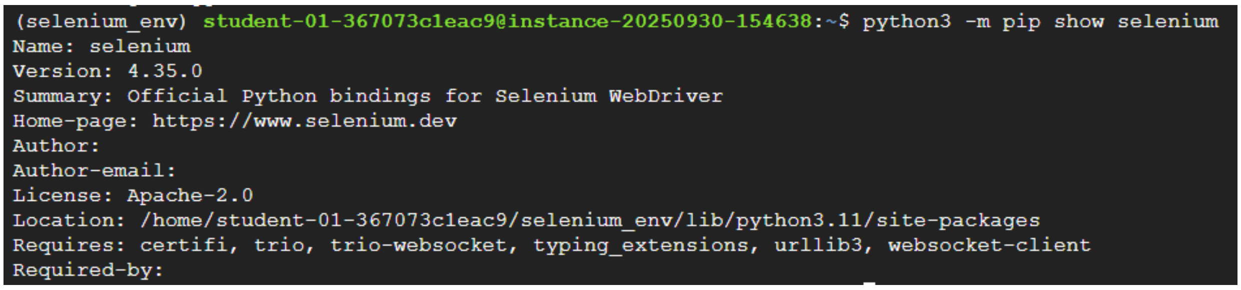
Check versions:
python3 --version
pip3 – – version



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ python3 --version
pip3 --version
Python 3.11.2
pip 25.2 from /home/student-01-367073c1eac9/selenium_env/lib/python3.11/site-packages/pip (python 3.11)
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$
```

4. Install Selenium: sudo apt install python3-selenium

Check installation:
python3 -m pip show selenium



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ python3 -m pip show selenium
Name: selenium
Version: 4.35.0
Summary: Official Python bindings for Selenium WebDriver
Home-page: https://www.selenium.dev
Author:
Author-email:
License: Apache-2.0
Location: /home/student-01-367073c1eac9/selenium_env/lib/python3.11/site-packages
Requires: certifi, trio, trio-websocket, typing_extensions, urllib3, websocket-client
Required-by:
```

Installing the Venv package
sudo apt install python3.11-venv

Create a virtual environment
python3 -m venv selenium_env

Whenever you want to use Selenium:
source selenium_env/bin/activate

Deactivate when done:
deactivate

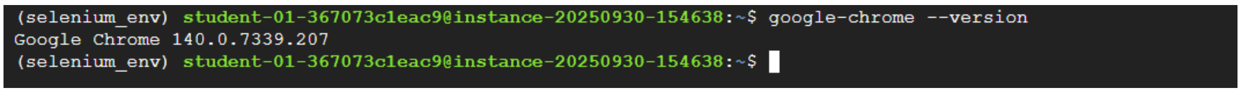
5. Install a WebDriver
Selenium needs a driver (like ChromeDriver or GeckoDriver) to interact with browsers.

Install Chrome:

wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb sudo apt install -y ./google-chrome-stable_current_amd64.deb

Verify:

google-chrome --version



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ google-chrome --version
Google Chrome 140.0.7339.207
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$
```

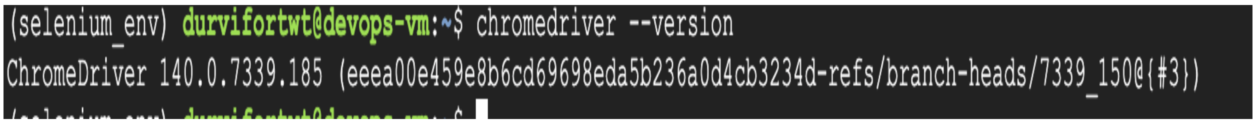
Install dependencies (Optional)
sudo apt install -y wget unzip curl xvfb libxi6 libgconf-2-4

Get latest ChromeDriver version
CHROME_DRIVER_VERSION=$(curl –sS chromedriver.storage.googleapis.com/LATEST_RELEASE)

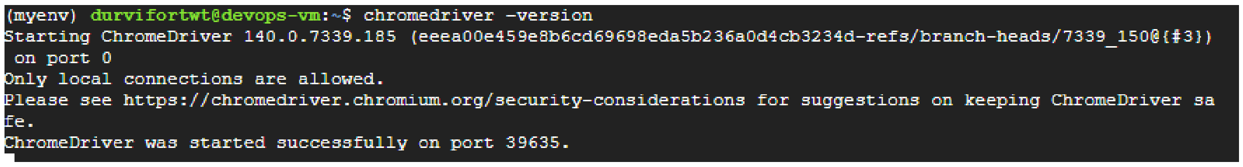
Download and install ChromeDriver

CHROME_VERSION=$(google-chrome --version | awk '{print $3}' | cut -d. -f1) wget https://chromedriver.storage.googleapis.com/${CHROME_VERSION}.0.5359.71/chromedriver_linux64.zip unzip chromedriver_linux64.zip sudo mv chromedriver /usr/local/bin/

Verify:
chromedriver –version



```
(selenium_env) durvifortwt@devops-vm:~$ chromedriver --version
ChromeDriver 140.0.7339.185 (eeea00e459e8b6cd69698eda5b236a0d4cb3234d-refs/branch-heads/7339_150@{#3})
```



```
(myenv) durvifortwt@devops-vm:~$ chromedriver –version
Starting ChromeDriver 140.0.7339.185 (eeea00e459e8b6cd69698eda5b236a0d4cb3234d-refs/branch-heads/7339_150@{#3})
 on port 0
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver sa
fe.
ChromeDriver was started successfully on port 39635.
```

It means:

- **ChromeDriver is installed** in your VM (chromedriver --version works).
- Its version is **140.0.7339.185**, which matches your installed **Google Chrome 140.x**.
- So Selenium will now be able to control Chrome without version mismatch errors.

6. Test Selenium

Create a test script (test_selenium.py):
nano test_selenium.py

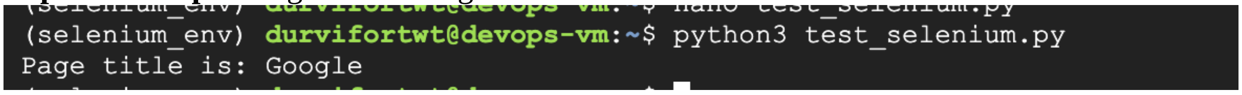
Paste this code:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By

# Point to installed ChromeDriver
service = Service("/usr/bin/chromedriver")

# Configure Chrome
options = webdriver.ChromeOptions()
options.add_argument("--headless")  # run without GUI (important on GCP VM)
options.add_argument("--no-sandbox")
options.add_argument("--disable-dev-shm-usage")

# Start WebDriver
driver = webdriver.Chrome(service=service, options=options)

# Open Google
driver.get("https://www.google.com")

# Print title
print("Page title is:", driver.title)
```

```
# Close browser
driver.quit()
```

Run it: python3 test_selenium.py

Expected Output: Page Title: Google



```
(selenium_env) durvifortwt@devops-vm:~$ nano test_selenium.py
(selenium_env) durvifortwt@devops-vm:~$ python3 test_selenium.py
Page title is: Google
```

This confirms:

- Python detects Selenium
- ChromeDriver is installed and working
- Browser automation runs headlessly in GCP

II. Write Selenium scripts for web UI testing.

General Structure of a Selenium Test Script

1. Set up WebDriver (headless mode in GCP).
2. Open the target website.
3. Locate web elements (input fields, buttons, links, etc.).
4. Perform actions (click, type, submit forms).
5. Validate results (page titles, messages, or elements).
6. Close the browser.

Example 1: Google Search Automation

- Create file: google_search.py
- Paste the code

```
from selenium import webdriver

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

import time
```

#Setup ChromeDriver

```
service = Service("/usr/bin/chromedriver")

options = webdriver.ChromeOptions()

options.add_argument("--headless") # required for GCP (no GUI)

options.add_argument("--no-sandbox")

options.add_argument("--disable-dev-shm-usage")

driver = webdriver.Chrome(service=service, options=options)
```

#Open Google

```
driver.get("https://www.google.com") print("Page title:", driver.title)
```

#Find search box and search for 'Selenium'

```
search_box = driver.find_element(By.NAME, "q")

search_box.send_keys("Selenium WebDriver")

search_box.send_keys(Keys.RETURN)

time.sleep(2) # wait for results to load
```

#Print the titles of search results

```
results = driver.find_elements(By.CSS_SELECTOR, "h3")

for i, result in enumerate(results[:5]):

# top 5 results

        print(f"{i+1}. {result.text}")

driver.quit()
```

Run it on GCP:

Lab experiment to be performed in this session:

1. Automate Google Search (basic navigation)

- Write a script that searches for a term (e.g., “Python Selenium tutorial”).
- Extract and print the first 5 search result titles.

- Validate that at least one result contains the word "Selenium".

Step 1: SSH into your VM

gcloud compute ssh instance-20250930-154638

Step 2: Update packages & install Python

sudo apt update && sudo apt upgrade -y

sudo apt install -y python3 python3-pip python3-venv wget unzip curl

Step 3: Create a Python virtual environment and activate it

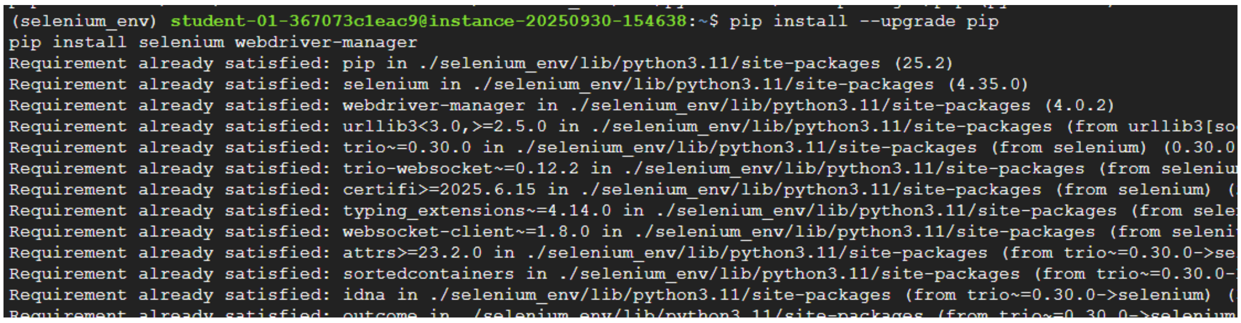
python3 -m venv selenium_env

source selenium_env/bin/activate

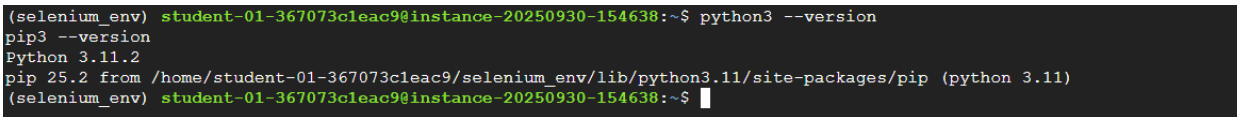
Step 4: Install Selenium and webdriver-manager

pip install --upgrade pip

pip install selenium webdriver-manager



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ pip install --upgrade pip
pip install selenium webdriver-manager
Requirement already satisfied: pip in ./selenium_env/lib/python3.11/site-packages (25.2)
Requirement already satisfied: selenium in ./selenium_env/lib/python3.11/site-packages (4.35.0)
Requirement already satisfied: webdriver-manager in ./selenium_env/lib/python3.11/site-packages (4.0.2)
Requirement already satisfied: urllib3<3.0,>=2.5.0 in ./selenium_env/lib/python3.11/site-packages (from urllib3[so
Requirement already satisfied: trio~=0.30.0 in ./selenium_env/lib/python3.11/site-packages (from selenium) (0.30.0
Requirement already satisfied: trio-websocket~=0.12.2 in ./selenium_env/lib/python3.11/site-packages (from seleniu
Requirement already satisfied: certifi>=2025.6.15 in ./selenium_env/lib/python3.11/site-packages (from selenium) (
Requirement already satisfied: typing_extensions~=4.14.0 in ./selenium_env/lib/python3.11/site-packages (from sele
Requirement already satisfied: websocket-client~=1.8.0 in ./selenium_env/lib/python3.11/site-packages (from seleni
Requirement already satisfied: attrs>=23.2.0 in ./selenium_env/lib/python3.11/site-packages (from trio~=0.30.0->se
Requirement already satisfied: sortedcontainers in ./selenium_env/lib/python3.11/site-packages (from trio~=0.30.0-
Requirement already satisfied: idna in ./selenium_env/lib/python3.11/site-packages (from trio~=0.30.0->selenium) (
Requirement already satisfied: outcome in ./selenium_env/lib/python3.11/site-packages (from trio~=0.30.0->selenium
```



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ python3 --version
pip3 --version
Python 3.11.2
pip 25.2 from /home/student-01-367073c1eac9/selenium_env/lib/python3.11/site-packages/pip (python 3.11)
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$
```

Step 5: Create the lab script

nano lab_google_search.py

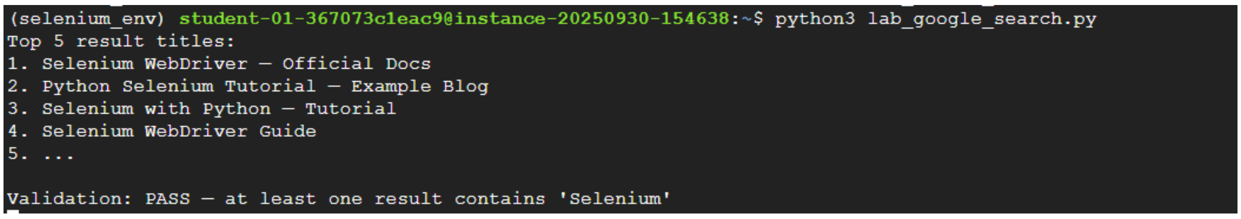
from selenium import webdriver

from selenium.webdriver.chrome.service import Service

```
from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from webdriver_manager.chrome import ChromeDriverManager

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected_conditions as EC

import sys


# Chrome options

options = webdriver.ChromeOptions()

options.add_argument("--headless=new")  # updated headless for latest Chrome

options.add_argument("--no-sandbox")

options.add_argument("--disable-dev-shm-usage")

options.add_argument("user-agent=Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/140 Safari/537.36")


# Setup ChromeDriver

service = Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service, options=options)


try:

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

    wait = WebDriverWait(driver, 10)

    wait.until(EC.presence_of_element_located((By.NAME, "q")))


    # Search for term

    search_box = driver.find_element(By.NAME, "q")

    term = "Python Selenium tutorial"
    search_box.send_keys(term)

    search_box.send_keys(Keys.RETURN)


    # Wait for results and get top 5 titles

    wait.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, "h3")))

    titles = [el.text for el in driver.find_elements(By.CSS_SELECTOR, "h3") if el.text.strip()]

    top5 = titles[:5]


    print("Top 5 result titles:")

    for i, t in enumerate(top5, 1):

        print(f"{i}. {t}")


    # Validate at least one contains "Selenium"

    if any("selenium" in t.lower() for t in top5):

        print("\nValidation: PASS — at least one result contains 'Selenium'")

    else:

        print("\nValidation: FAIL — none of the top 5 contains 'Selenium'")


finally:

    driver.quit()
```

Step 6: Run the script

python3 lab_google_search.py



```
(selenium_env) student-01-367073c1eac9@instance-20250930-154638:~$ python3 lab_google_search.py
Top 5 result titles:
1. Selenium WebDriver — Official Docs
2. Python Selenium Tutorial — Example Blog
3. Selenium with Python — Tutorial
4. Selenium WebDriver Guide
5. ...

Validation: PASS — at least one result contains 'Selenium'
```