**Aim:** Generate regression test case suite for the software.

**Description:**

**Selenium** is an open-source framework and suite of tools primarily used for automating web browser interactions. It is widely employed for testing web applications to ensure they function correctly across various browsers and platforms. Selenium supports multiple programming languages, including Python, Java, C#, Ruby, JavaScript, and more.

**Components of Selenium**

1. **Selenium WebDriver**
   * A core component that provides APIs to interact with web browsers.
   * Automates user actions like clicking buttons, filling forms, and navigating between pages.
   * Works with all major browsers (e.g., Chrome, Firefox, Edge, Safari).
2. **Selenium IDE**
   * A browser extension (for Chrome and Firefox) for recording and replaying test cases.
   * Ideal for beginners or quick prototyping.
3. **Selenium Grid**
   * Allows running tests on multiple browsers and platforms in parallel.
   * Useful for distributed testing and scaling up test automation.

**Key Features**

* **Cross-Browser Testing:** Supports popular browsers like Chrome, Firefox, Edge, Safari, and more.
* **Cross-Platform Support:** Works on Windows, macOS, and Linux.
* **Multiple Language Support:** Write test scripts in Python, Java, C#, JavaScript, etc.
* **Community-Driven:** Maintained by a strong open-source community.
* **Integration:** Easily integrates with tools like Jenkins, Maven, Docker, and TestNG for CI/CD workflows.

**Applications of Selenium**

1. **Automated Testing:**
   * Functional testing of web applications.
   * Regression testing for checking if changes have broken existing functionality.
2. **Web Scraping:** *(With caution for legal and ethical considerations)*
   * Automating data extraction from websites.
3. **Continuous Testing:**
   * Integrating with CI/CD pipelines to automate testing during deployment.

**Why Use Selenium?**

* **Free and Open Source:** No licensing cost.
* **Flexible:** Works with multiple browsers, platforms, and languages.
* **Extensible:** Can be extended with third-party tools or frameworks like TestNG, PyTest, and Cucumber.
* **Scalable:** Suitable for small projects and large-scale enterprise applications.

**Implementation:**

***index.html***

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>BMI Calculator</title>

<style>

body {

font-family: Arial, sans-serif;

background: linear-gradient(135deg, #a8e0ff, #c9e8ff);

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

margin: 0;

color: #333;

}

h1 {

color: #0056b3;

text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3);

}

label {

margin: 10px 0 5px;

font-weight: bold;

color: #007bff;

}

input[type="number"], select {

padding: 10px;

width: 200px;

border: 2px solid #007bff;

border-radius: 5px;

margin-bottom: 20px;

transition: border-color 0.3s;

}

input[type="number"]:focus, select:focus {

border-color: #0056b3;

outline: none;

}

button {

padding: 10px 15px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s;

}

button:hover {

background-color: #218838;

}

h2 {

margin-top: 20px;

color: #dc3545; /\* Red for result text \*/

}

.bmi-chart {

margin-top: 20px;

text-align: left;

color: #333;

}

.bmi-chart span {

font-weight: bold;

color: #007bff;

}

</style>

</head>

<body>

<h1>BMI Calculator</h1>

<label for="weight">Weight (kg):</label>

<input type="number" id="weight" required>

<br>

<label for="height">Height (m):</label>

<input type="number" id="height" step="0.01" required>

<br>

<label for="age">Age (years):</label>

<input type="number" id="age" required>

<br>

<label for="gender">Gender:</label>

<select id="gender">

<option value="male">Male</option>

<option value="female">Female</option>

</select>

<br>

<button onclick="calculateBMI()">Calculate BMI</button>

<h2 id="result"></h2>

<div class="bmi-chart" id="bmi-chart" style="display: none;">

<h3>BMI Categories:</h3>

<p><span>Underweight:</span> Less than 18.5</p>

<p><span>Normal weight:</span> 18.5 – 24.9</p>

<p><span>Overweight:</span> 25 – 29.9</p>

<p><span>Obesity:</span> 30 or greater</p>

</div>

<script>

function calculateBMI() {

const weight = parseFloat(document.getElementById('weight').value);

const height = parseFloat(document.getElementById('height').value);

const age = parseInt(document.getElementById('age').value);

const gender = document.getElementById('gender').value;

if (weight > 0 && height > 0 && age > 0) {

const bmi = weight / (height \* height);

const category = getBMICategory(bmi);

document.getElementById('result').innerText =

`Your BMI is: ${bmi.toFixed(2)} (${category})`;

// Show BMI chart

document.getElementById('bmi-chart').style.display = 'block';

} else {

document.getElementById('result').innerText =

'Please enter valid weight, height, and age.';

}

}

function getBMICategory(bmi) {

if (bmi < 18.5) return 'Underweight';

else if (bmi >= 18.5 && bmi < 24.9) return 'Normal weight';

else if (bmi >= 25 && bmi < 29.9) return 'Overweight';

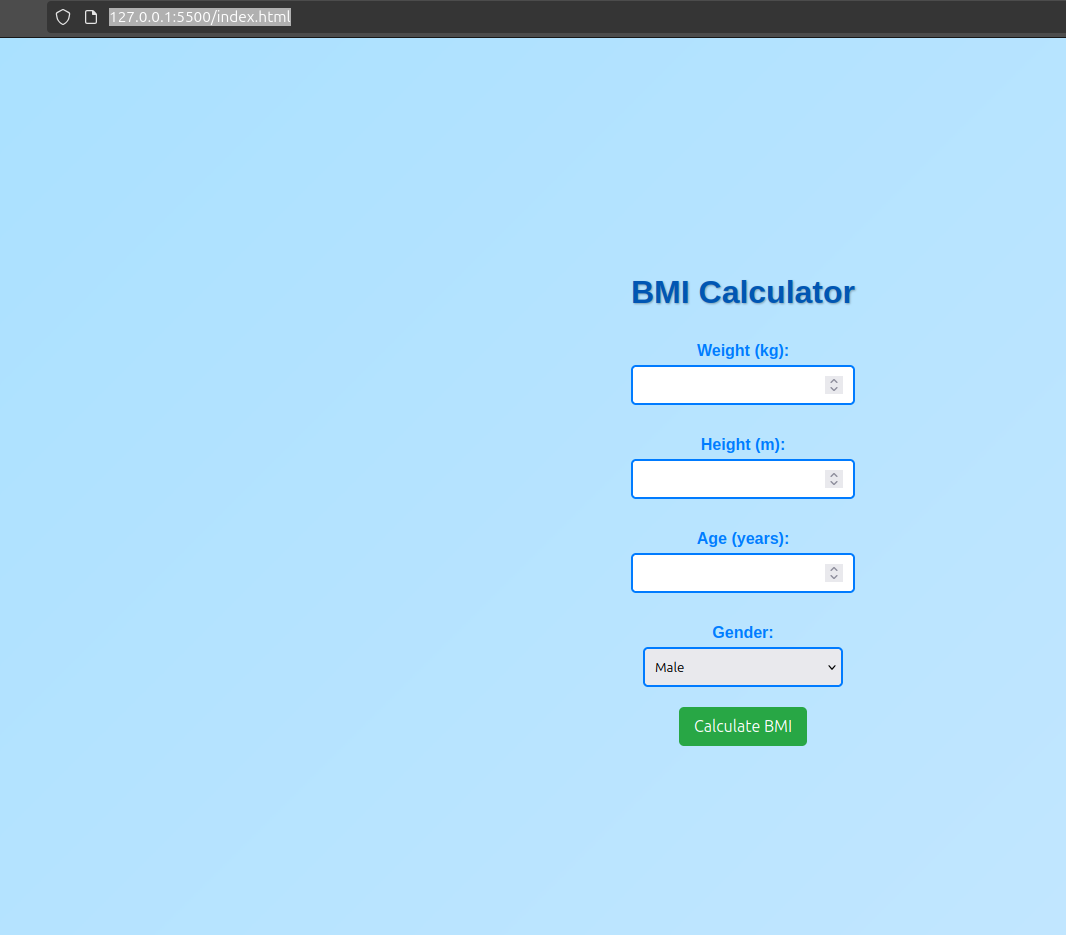
else return 'Obesity';

}

</script>

</body>

</html>



***test\_defaultSuite.py***

# Generated by Selenium IDE

import pytest

import time

import json

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.action\_chains import ActionChains

from selenium.webdriver.support import expected\_conditions

from selenium.webdriver.support.wait import WebDriverWait

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.desired\_capabilities import DesiredCapabilities

class TestDefaultSuite():

def setup\_method(self, method):

self.driver = webdriver.Firefox()

self.vars = {}

def teardown\_method(self, method):

self.driver.quit()

def test\_testcase1(self):

self.driver.get("http://127.0.0.1:5500/index.html")

self.driver.set\_window\_size(550, 691)

self.driver.find\_element(By.ID, "weight").click()

self.driver.find\_element(By.ID, "weight").send\_keys("57")

self.driver.find\_element(By.ID, "height").click()

self.driver.find\_element(By.ID, "height").send\_keys("1.53")

self.driver.find\_element(By.ID, "age").click()

self.driver.find\_element(By.ID, "age").send\_keys("25")

self.driver.find\_element(By.ID, "gender").click()

dropdown = self.driver.find\_element(By.ID, "gender")

dropdown.find\_element(By.XPATH, "//option[. = 'Female']").click()

self.driver.find\_element(By.CSS\_SELECTOR, "option:nth-child(2)").click()

self.driver.find\_element(By.CSS\_SELECTOR, "button").click()

***test\_bmi\_calculator.py***

from selenium import webdriver # type: ignore

from selenium.webdriver.firefox.service import Service # type: ignore

from selenium.webdriver.common.by import By # type: ignore

import time

# Set up the WebDriver (replace with the path to your driver if needed)

service = Service(executable\_path='/home/sysadmin/Downloads/geckodriver') # Change this to your actual path

driver = webdriver.Firefox()

profile\_path = '/home/sysadmin/snap/firefox/common/.mozilla/firefoz/z59yv691.GeetaSeshapalli'

#driver = webdriver.Firefox(executable\_path='/home/sysadmin/Downloads/geckodriver') # Use 'webdriver.Firefox()' for Firefox

try:

# Step 1: Open the HTML file

driver.get("/home/sysadmin/Desktop/SW4/") # Update with the correct path

# Step 2: Find the weight, height, age, and gender input fields

weight\_input = driver.find\_element(By.ID, "weight")

height\_input = driver.find\_element(By.ID, "height")

age\_input = driver.find\_element(By.ID, "age")

gender\_select = driver.find\_element(By.ID, "gender")

# Step 3: Enter weight, height, age, and select gender

weight\_input.send\_keys("70") # Example weight in kg

height\_input.send\_keys("1.75") # Example height in meters

age\_input.send\_keys("25") # Example age

gender\_select.send\_keys("Male") # Select gender

# Step 4: Click the Calculate button

calculate\_button = driver.find\_element(By.XPATH, "//button[text()='Calculate BMI']")

calculate\_button.click()

# Step 5: Wait for the result to appear

time.sleep(1) # Wait for a moment to ensure the result is displayed

# Step 6: Get the result text

result = driver.find\_element(By.ID, "result").text

print(result) # Output the result to console

# Step 7: Optionally, check if the BMI category is correct

if "Obesity" in result or "Overweight" in result:

print("BMI category is correctly identified.")

else:

print("Check the BMI category.")

finally:

# Close the browser after testing

driver.quit()

*Here’s a step-by-step guide to generate automated test cases using the* ***Selenium IDE Extension****:*

***1. Install Selenium IDE Extension***

1. ***Google Chrome:***
   * *Go to the Chrome Web Store.*
   * *Search for "Selenium IDE" and click* ***Add to Chrome****.*
2. ***Mozilla Firefox:***
   * *Go to the* [*Mozilla Add-ons page*](https://addons.mozilla.org)*.*
   * *Search for "Selenium IDE" and click* ***Add to Firefox****.*

***2. Launch Selenium IDE***

1. *Open the browser where you installed the extension.*
2. *Click on the Selenium IDE icon in the browser toolbar.*
3. *Start a new project:*
   * *Click* ***Create a New Project*** *and provide a name.*

***3. Record a Test***

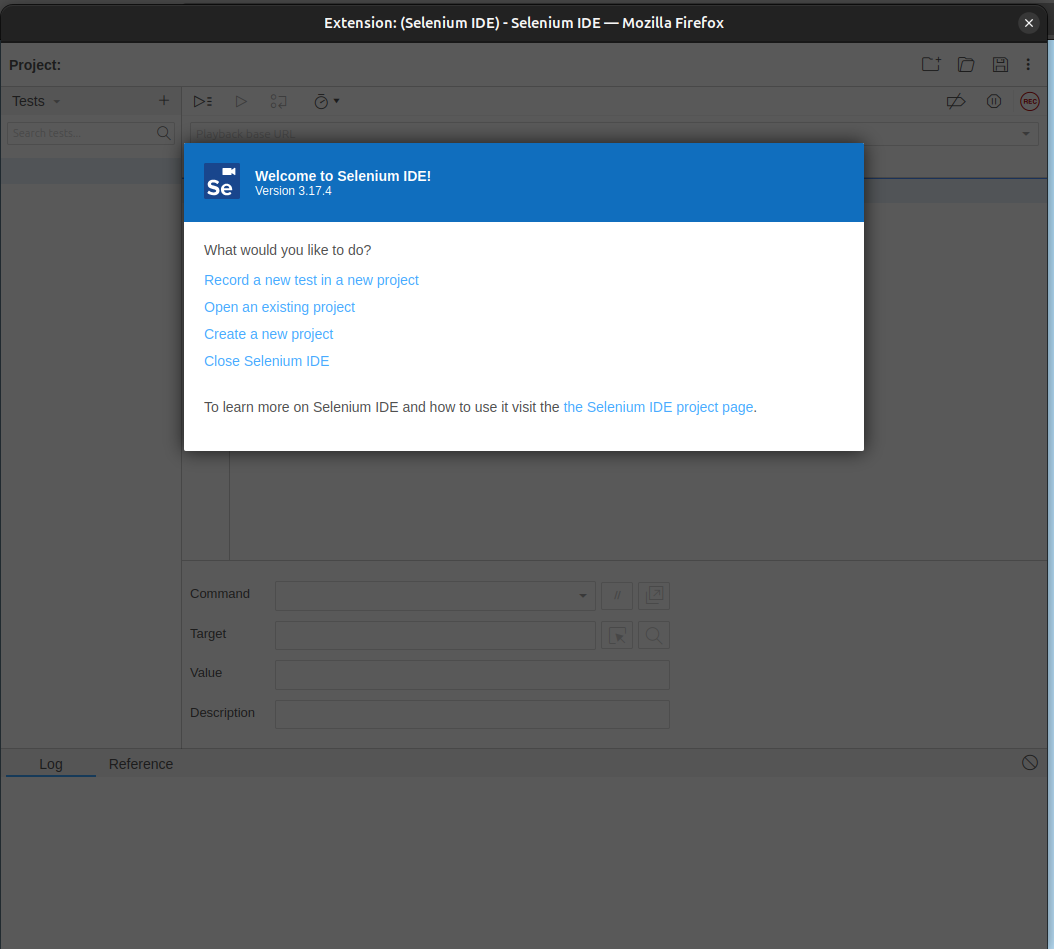
1. ***Start Recording:***
   * *Click the* ***Record a New Test in a New Project*** *button.*
   * *Enter the base URL of the website you want to test (e.g., https://www.google.com).*
   * *Selenium IDE will open a new tab for recording.*
2. ***Perform Actions:***
   * *Interact with the web application (e.g., clicking buttons, filling forms, navigating pages). Selenium IDE will record these actions as steps in the test case.*
3. ***Stop Recording:***
   * *When finished, return to the Selenium IDE tab and click the* ***Stop Recording*** *button.*
   * *Name your test case and save it.*

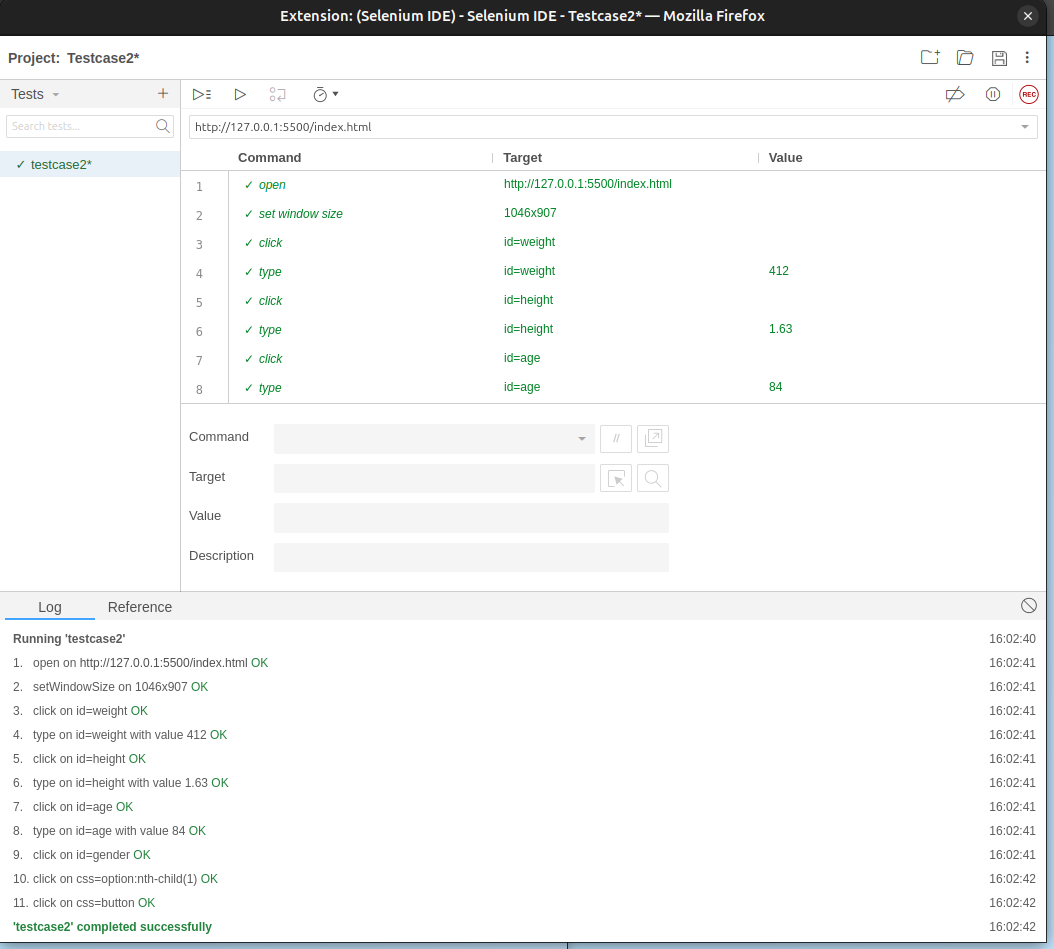
***4. Run the Test***

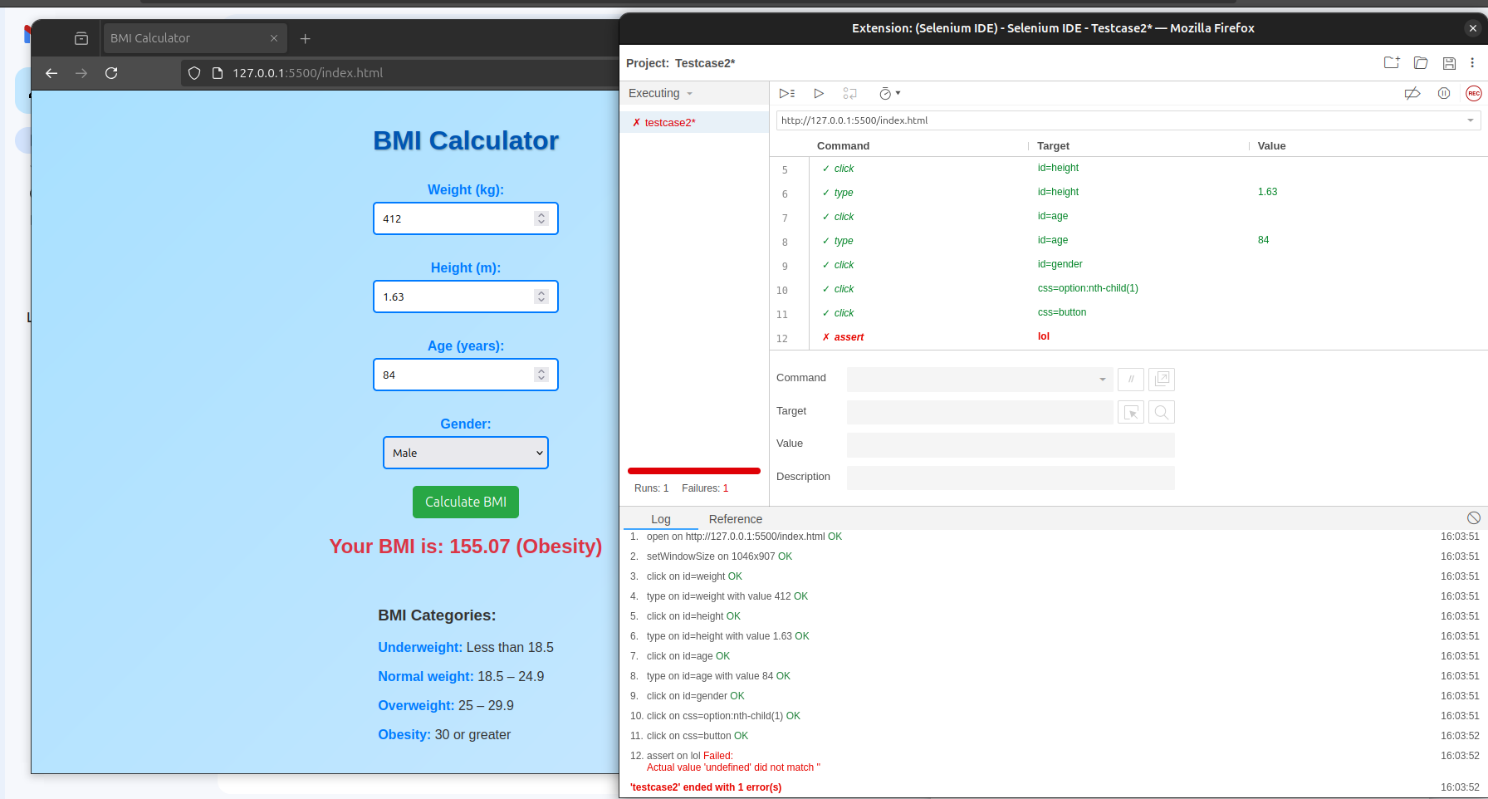
1. *Select the test case in Selenium IDE.*
2. *Click the* ***Run Current Test*** *button (▶️).*
3. *Observe the test execution in a browser window.*

***5. Export the Test***

1. *Click the* ***Export*** *button in Selenium IDE.*
2. *Select the format for export. For Python, choose* ***Python - Selenium WebDriver****.*
3. *Save the file to your system.*







**Conclusion:**

Using the **Selenium IDE extension** for generating automated test cases provides a user-friendly and efficient way to create, execute, and manage tests for web applications. It simplifies the automation process with the following advantages:

1. **Ease of Use**: Selenium IDE offers an intuitive interface for recording and managing test cases without requiring in-depth programming knowledge.
2. **Customizability**: Recorded tests can be exported to various programming languages (e.g., Python, Java) for further customization and integration with advanced frameworks.
3. **Time Efficiency**: Recording and playback reduce the time spent on writing repetitive test scripts.
4. **Flexibility**: Tests can be run directly in the IDE or exported and executed in CI/CD pipelines for continuous testing.
5. **Scalability**: Recorded tests can serve as a foundation for building comprehensive automated testing suites for complex applications.

By leveraging Selenium IDE, teams can quickly generate and maintain test cases, leading to more efficient testing cycles and higher software quality.