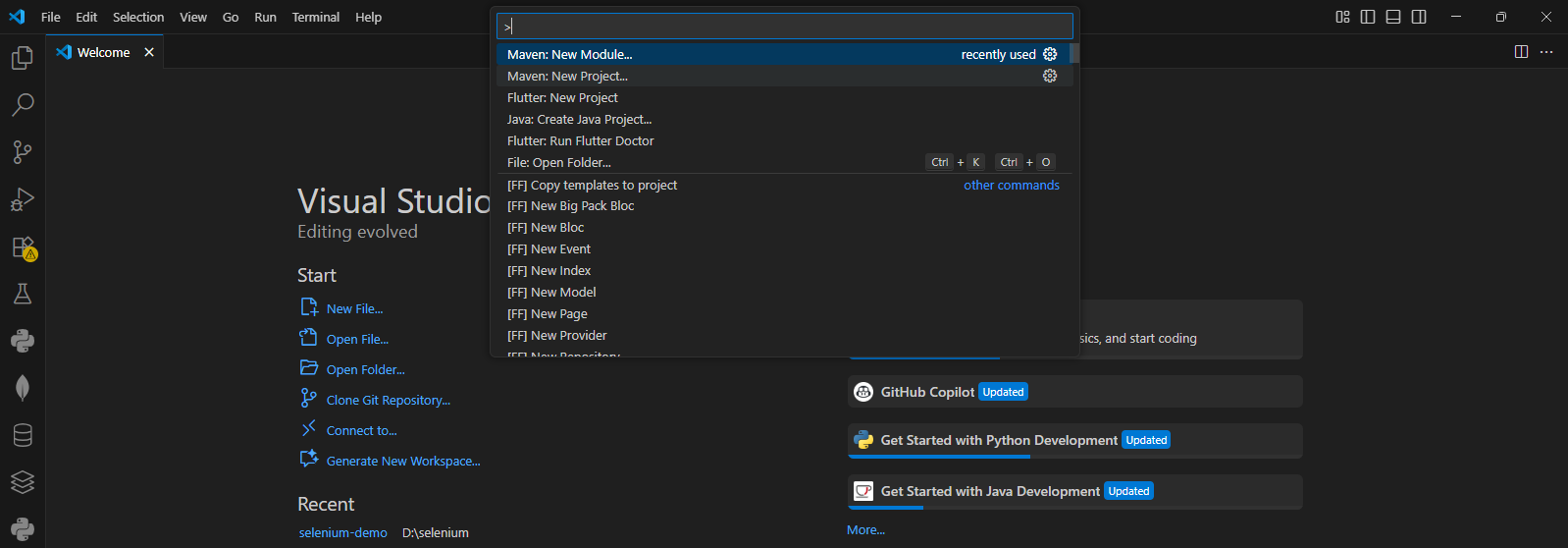
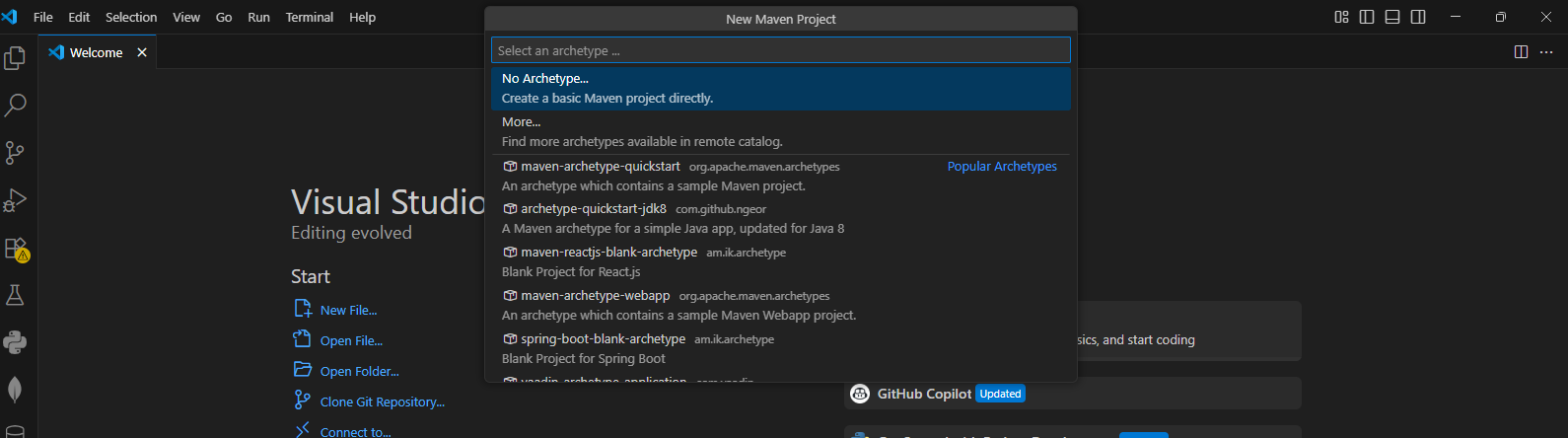
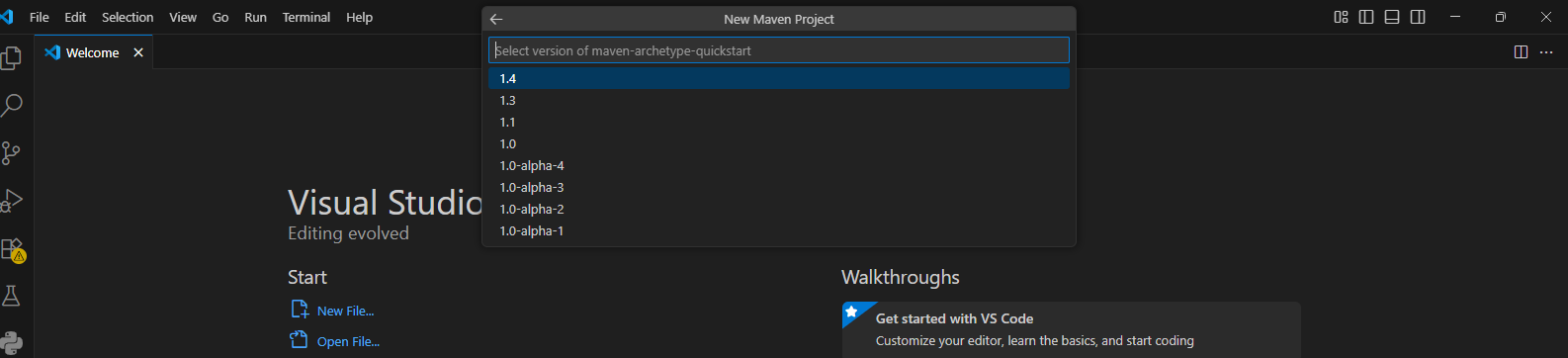
**WEEK-11: Write a simple program in JavaScript and perform testing using Selenium.**

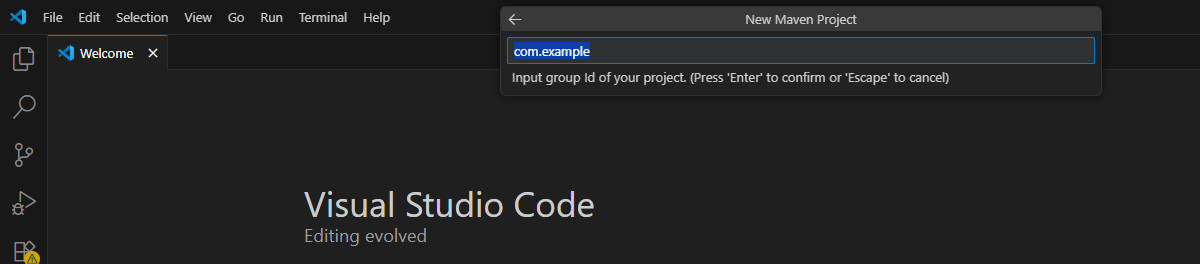
### STEP 1: Create a Maven Project in VS Code

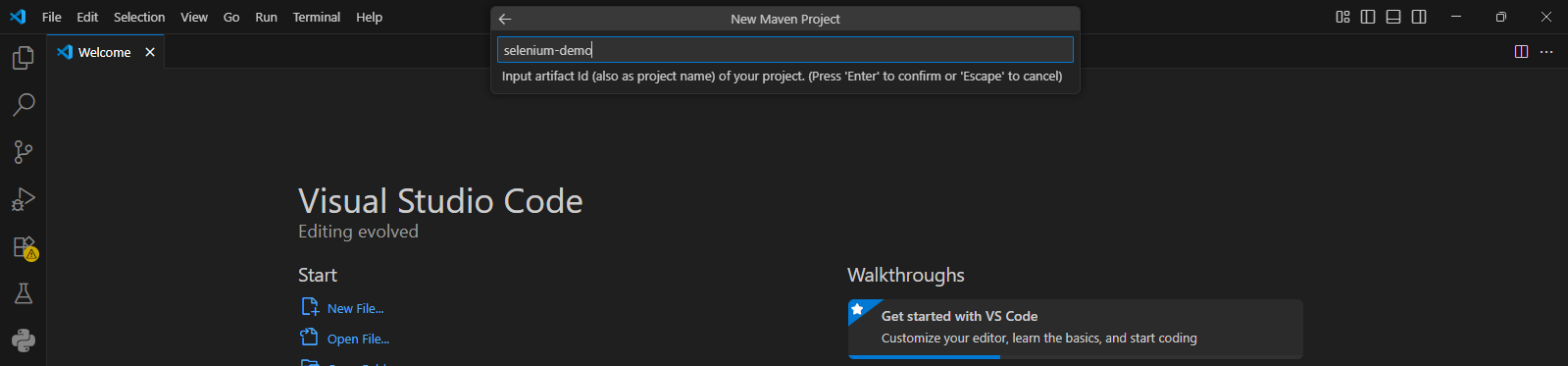
1. **Open VS Code**.
2. Press Ctrl+Shift+P to open the **Command Palette**.
3. Search for and select Maven: Create Maven Project.
4. Select archetype: **maven-archetype-quickstart.**
5. Select version: 1.4.
6. Enter **Group ID**: com.example
7. Enter **Artifact ID**: selenium-demo
8. Choose a folder to save the project and click **Finish**.
9. VS Code will generate the Maven project structure.

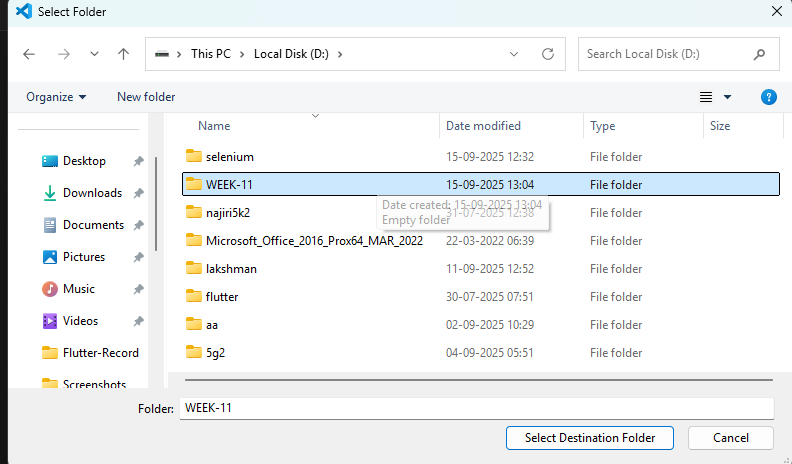
****

****

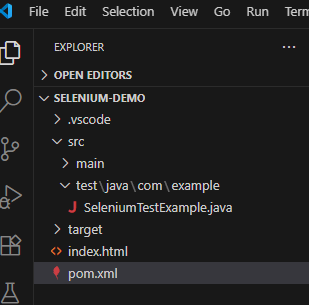
****

****

****

****

**Project Structure:**

****

### STEP 2: Add index.html File

1. In the root folder of your Maven project, **create a new file** named **index.html.**
2. Paste the following HTML code into the file:

**index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Simple JS Example</title>

</head>

<body>

<h1 id="heading">Original Heading</h1>

<button id="btn" onclick="changeText()">Click Me</button>

<script>

function changeText() {

document.getElementById("heading").innerText = "Hello, Selenium!";

}

</script>

</body>

</html>

3. Save the file.

### STEP 3: Create Selenium Test Class

1. In the Maven structure, navigate to: **src/test/java/com/example/**
2. Create a new Java file: **SeleniumTestExample.java**

**SeleniumTestExample.java**

package com.example;

import org.junit.jupiter.api.AfterAll;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import io.github.bonigarcia.wdm.WebDriverManager;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import java.nio.file.Paths;

public class SeleniumTestExample {

private static WebDriver driver;

@BeforeAll

static void setUp() {

WebDriverManager.chromedriver().setup();

ChromeOptions options = new ChromeOptions();

// Uncomment to run headless (useful on CI):

// options.addArguments("--headless=new"); // or "--headless" for older Chrome

driver = new ChromeDriver(options);

}

@AfterAll

static void tearDown() {

if (driver != null) {

driver.quit();

}

}

@Test

void testButtonClickChangesHeading() {

String filePath = Paths.get("index.html").toAbsolutePath().toUri().toString();

driver.get(filePath);

WebElement heading = driver.findElement(By.id("heading"));

assertEquals("Original Heading", heading.getText(), "Initial heading should match");

driver.findElement(By.id("btn")).click();

// Re-locate heading after click

heading = driver.findElement(By.id("heading"));

assertEquals("Hello, Selenium!", heading.getText(), "Heading should update after click");

}

}

### STEP 4: Add Required Dependencies in pom.xml

In your Maven project, open the pom.xml file and add these dependencies:

**pom.xml**

<dependencies>

<!-- JUnit Jupiter (JUnit 5) -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<!-- Selenium Java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.21.0</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.12</version>

<scope>test</scope>

</dependency>

<!-- WebDriverManager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.8.0</version>

</dependency>

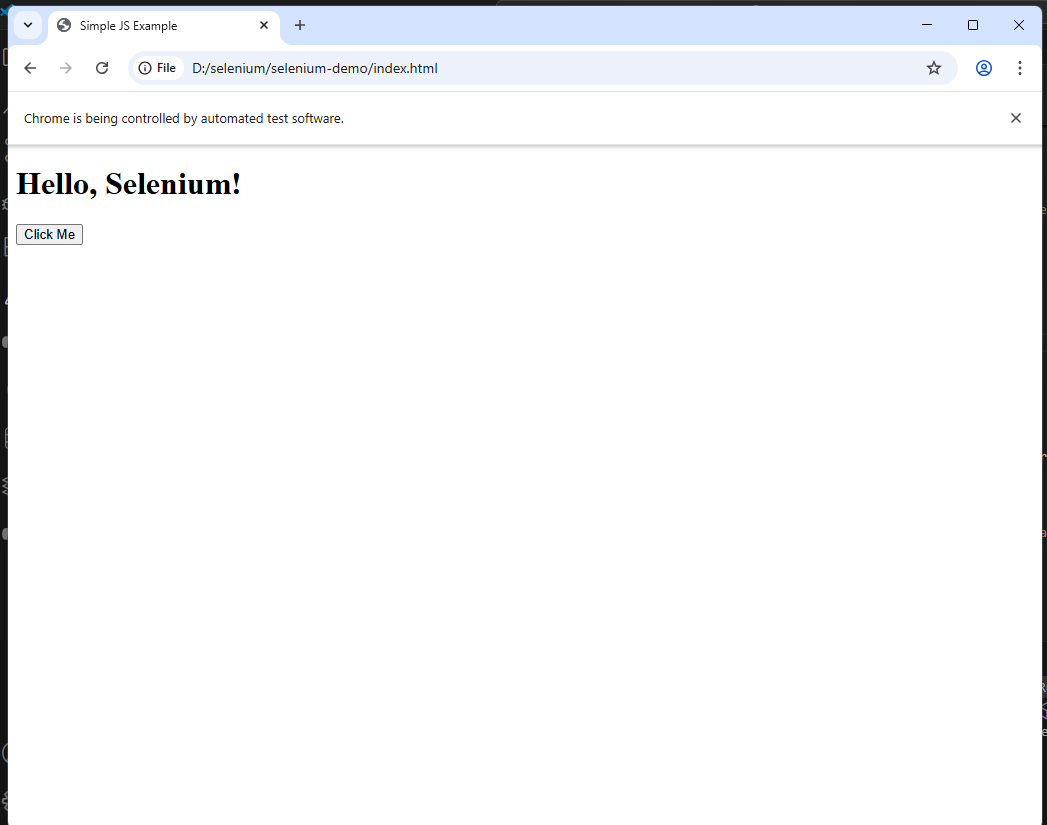
</dependencies>

### STEP 5: Run the Selenium Test

1. Open the **Terminal** in VS Code.
2. Navigate to your Maven project folder (if not already there).
3. Run the following command: **mvn test**

You should see output indicating that the test ran and passed.

✅ It will open Chrome, load your index.html, click the button, and verify the heading changes to: **"Hello, Selenium!"**



**12. Develop test cases for the above containerized application using selenium.**

### STEP 1: Create a Maven Project in VS Code

1. **Open VS Code**.
2. Press Ctrl+Shift+P to open the **Command Palette**.
3. Search for and select Maven: Create Maven Project.
4. Select archetype: **maven-archetype-quickstart.**
5. Select version: 1.4.
6. Enter **Group ID**: com.registration
7. Enter **Artifact ID**: registration-test
8. Choose a folder to save the project and click **Finish**.
9. VS Code will generate the Maven project structure.

registration-test/

├── src/

│ ├── main/java/com/registration/App.java

│ └── test/java/com/registration/AppTest.java

├── pom.xml

### ****STEP 2: Replace the**** pom.xml ****with Dependencies for Selenium****

Replace the contents of pom.xml with the following:

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>selenium-demo</artifactId>

  <version>1.0-SNAPSHOT</version>

  <properties>

    <maven.compiler.source>21</maven.compiler.source>

    <maven.compiler.target>21</maven.compiler.target>

    <junit.jupiter.version>5.10.2</junit.jupiter.version>

    <selenium.version>4.25.0</selenium.version>

    <webdrivermanager.version>5.9.2</webdrivermanager.version>

  </properties>

  <dependencies>

    <!-- Selenium -->

    <dependency>

      <groupId>org.seleniumhq.selenium</groupId>

      <artifactId>selenium-java</artifactId>

      <version>${selenium.version}</version>

    </dependency>

    <!-- WebDriverManager -->

    <dependency>

      <groupId>io.github.bonigarcia</groupId>

      <artifactId>webdrivermanager</artifactId>

      <version>${webdrivermanager.version}</version>

    </dependency>

    <!-- JUnit Jupiter -->

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter</artifactId>

      <version>${junit.jupiter.version}</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Maven Surefire Plugin for JUnit 5 -->

      <plugin>

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-surefire-plugin</artifactId>

        <version>3.2.5</version>

        <configuration>

          <useModulePath>false</useModulePath>

        </configuration>

      </plugin>

    </plugins>

  </build>

</project>

**STEP 3: Update the Test Class**

**AppTest.java**

package com.registration;

import io.github.bonigarcia.wdm.WebDriverManager;

import org.junit.jupiter.api.\*;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

public class AppTest {

private WebDriver driver;

@BeforeEach

public void setUp() {

WebDriverManager.chromedriver().setup();

driver = new ChromeDriver();

}

@Test

@Order(1)

public void testFormSubmission() {

driver.get("http://localhost:8080");

// Fill form

driver.findElement(By.id("name")).sendKeys("vinay");

driver.findElement(By.id("email")).sendKeys("vinay@gmail.com");

driver.findElement(By.id("phone")).sendKeys("1234567890");

WebElement gender = driver.findElement(By.id("gender"));

gender.sendKeys(“Male");

WebElement event = driver.findElement(By.id("event"));

event.sendKeys("coding contest");

driver.findElement(By.cssSelector("button[type='submit']")).click();

// Assert page contains form title again (no backend to verify success)

Assertions.assertTrue(driver.getPageSource().contains("Event Registration"));

}

@AfterEach

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

STEP 4: Update the HTML (index.html) File

Make sure your index.html inside the Docker project contains this corrected version:

<form action="submit" method="POST">

<h2>Event Registration</h2>

<label for="name">Full Name</label>

<input type="text" id="name" name="name" required><br>

<label for="email">Email Address</label>

<input type="email" id="email" name="email" required><br>

<label for="phone">Phone Number</label>

<input type="tel" id="phone" name="phone" required><br>

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

<select id="gender" name="gender">

<option>Male</option>

<option>Female</option>

<option>Other</option>

</select>

<label for="event">Choose Event</label>

<select id="event" name="event">

<option value="coding">Coding Contest</option>

<option value="debate">Debate</option>

<option value="music">Music Show</option>

</select>

<button type="submit">Register</button>

</form>

**STEP 5: Rebuild and Run the Docker Container**

In the folder containing your Dockerfile and index.html:

docker build -t registration-form .

docker run -dp 8080:80 registration-form

Make sure it's accessible at:

➡️ <http://localhost:8080>

**STEP 6: Run the Selenium Test**

In VS Code terminal or command prompt: **mvn test**

