package hellocucumber;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import io.appium.java\_client.android.AndroidDriver;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import io.cucumber.spring.CucumberContextConfiguration;

import org.junit.runner.RunWith;

import org.openqa.selenium.\*;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.io.FileHandler;

import org.openqa.selenium.remote.DesiredCapabilities;

import org.openqa.selenium.remote.RemoteWebDriver;

import java.io.File;

import java.io.IOException;

import java.net.URL;

import java.time.LocalDateTime;

import java.util.Base64;

import java.util.List;

import java.util.Objects;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or

// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.

@CucumberContextConfiguration

@RunWith(Cucumber.class)

@CucumberOptions(

// features should start with "src/test/resources/features/xxxxxxxxxxx.feature",

features = "src/test/resources/features/Selenium\_Android.feature",

// glue starts with "src/test/java/#package for the Test Java class with the Given;When;Then annotations#/",

glue={"src/test/java/hellocucumber/"},

// plugin = = {"pretty", "html:target/cucumber-reports"}

plugin = {"pretty", "html:target/cucumber-reports"}

)

public class Test {

// Test.java class should be within src > test

AndroidDriver driver1;

AndroidDriver driver2;

AndroidDriver driver3;

AndroidDriver driver4;

String websiteTitle = "";

String capturedScreenshotImageFilepathString = "";

String currentLocalDateTimeForExtentSparkReporter = *currentLocalDateTimeWithDdMmYyFormat*();

ExtentReports extentReport = new ExtentReports();

// should be target/Spark.html, not an actual filepath starting from C:\\

ExtentSparkReporter extentSparkReporter = new ExtentSparkReporter("target/Spark\_" + currentLocalDateTimeForExtentSparkReporter + ".html");

ExtentTest extentTest = extentReport.createTest("Selenium\_Android Test");

public Test() throws IOException {

}

// Note that this method is not executed, if running the feature with the Given, When, Then lines

void testMethod() {

System.*out*.println("testing the IntelliJ IDEA - Help - Testing - Create Tests - Right-click to generate Test Method for JUnit5.");

}

@org.junit.Test

@org.junit.jupiter.api.Test

@Given("that the easiest way to find WebElements is by driver.findElement; and driver.findElements which returns a List WebElement")

public void testMethod1() throws IOException {

// this line only works without an error, in a public method, as it is a public void method

// the other three lines (originally above this fourth line) are public methods

extentReport.attachReporter(extentSparkReporter);

DesiredCapabilities desiredCapabilities = new DesiredCapabilities();

desiredCapabilities.setCapability("platformName", "Android");

desiredCapabilities.setCapability("appium:platformVersion", "15.0");

desiredCapabilities.setCapability("appium:app", "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\com.afwsamples.testdpc\_9.0.12-9012\_minAPI21(nodpi)\_apkmirror.com.apk");

desiredCapabilities.setCapability("appium:automationName", "UiAutomator2");

driver1 = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"), desiredCapabilities);

WebElement policyManagementText = driver1.findElement(By.*xpath*("//android.widget.TextView[@text=\"Policy management\"]"));

if(policyManagementText.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver1);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "Able to find \"Policy Management\".");

System.*out*.println("Pass");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver1);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "Unable to find \"Policy Management\".");

System.*out*.println("Fail");

}

}

@org.junit.Test

@org.junit.jupiter.api.Test

@When("it is not possible to do so for certain WebElements in this way")

public void testMethod2 () throws IOException {

WebElement policyManagementText = driver1.findElement(By.*xpath*("//android.widget.TextView[@text=\"Policy management\"]"));

if(policyManagementText.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver1);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "Able to find \"Policy Management\".");

System.*out*.println("Pass");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver1);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "Unable to find \"Policy Management\".");

System.*out*.println("Fail");

}

}

@org.junit.Test

@org.junit.jupiter.api.Test

@Then("we use driver.findElement By.tagName\"...\", for which ... can be i for <i> or button for <button>")

public void testMethod3 () throws IOException {

DesiredCapabilities desiredCapabilities = new DesiredCapabilities();

desiredCapabilities.setCapability("platformName", "Android");

desiredCapabilities.setCapability("appium:platformVersion", "15.0");

desiredCapabilities.setCapability("appium:app", "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\ApiDemos-debug.apk");

desiredCapabilities.setCapability("appium:automationName", "UiAutomator2");

driver2 = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"), desiredCapabilities);

WebElement appTab = driver2.findElement(By.*xpath*("//android.widget.TextView[@content-desc=\"App\"]"));

if(appTab.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "App tab is displayed.");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "App tab is not displayed.");

}

appTab.click();

WebElement searchTab = driver2.findElement(By.*xpath*("//android.widget.TextView[@content-desc=\"Search\"]"));

if(searchTab.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "Search tab is displayed.");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "Search tab is not displayed.");

}

searchTab.click();

WebElement invokeSearchTab = driver2.findElement(By.*xpath*("//android.widget.TextView[@content-desc=\"Invoke Search\"]"));

if(invokeSearchTab.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "Invoke Search tab is displayed.");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "Invoke Search tab is not displayed.");

}

invokeSearchTab.click();

WebElement prefillQueryTextField = driver2.findElement(By.*xpath*("//android.widget.EditText[@resource-id=\"io.appium.android.apis:id/txt\_query\_prefill\"]"));

prefillQueryTextField.sendKeys("Test the text.");

if(prefillQueryTextField.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "Able to enter the text \"Test the text.\".");

System.*out*.println("Pass");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver2);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "Unable to enter the text \"Test the text.\".");

System.*out*.println("Fail");

}

desiredCapabilities = new DesiredCapabilities();

desiredCapabilities.setCapability("platformName", "Android");

desiredCapabilities.setCapability("appium:platformVersion", "15.0");

desiredCapabilities.setCapability("appium:app", "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\com.microsoft.emmx.canary\_138.0.3336.0-333600000\_minAPI26(armeabi-v7a)(nodpi)\_apkmirror.com.apk");

desiredCapabilities.setCapability("appium:automationName", "UiAutomator2");

driver3 = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"), desiredCapabilities);

WebElement edgeBrowserDefaultWebpageSearchBar = driver3.findElement(By.*xpath*("//android.widget.EditText[@resource-id=\"com.microsoft.emmx.canary:id/url\_bar\"]"));

if(edgeBrowserDefaultWebpageSearchBar.isDisplayed()){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver3);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Edge Browser default webpage search bar is displayed.");

System.*out*.println("Pass");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver3);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "he Edge Browser default webpage search bar is not displayed.");

System.*out*.println("Fail");

}

// Note, no text was entered, but these two following lines did not cause any error

edgeBrowserDefaultWebpageSearchBar.sendKeys("MapleStorySEA Unfunded Amino");

edgeBrowserDefaultWebpageSearchBar.sendKeys(Keys.*ENTER*);

// So this last screenshot image is the default webpage without any UI changes

String oneString = "1";

if(Objects.*equals*(oneString,"1")){

capturedScreenshotImageFilepathString = *takeScreenshot*(driver3);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Edge Browser default webpage search bar is displayed.");

System.*out*.println("Pass");

} else {

capturedScreenshotImageFilepathString = *takeScreenshot*(driver3);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "he Edge Browser default webpage search bar is not displayed.");

System.*out*.println("Fail");

}

extentReport.flush();

}

// This method is for the filename for capturing screenshot images which are not in the ExtentReport

public static String takeScreenshot(WebDriver driver) throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

TakesScreenshot screenshot = ((TakesScreenshot) driver);

File sourceFile = screenshot.getScreenshotAs(OutputType.*FILE*);

File destinationFile = new File("C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium Screenshot Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png");

FileHandler.*copy*(sourceFile, destinationFile);

return "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium Screenshot Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png";

}

// This method is for returning the substring of the current LocalDateTome, for the filename of the ExtentReport

public static String currentLocalDateTimeWithDdMmYyFormat() throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Android\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

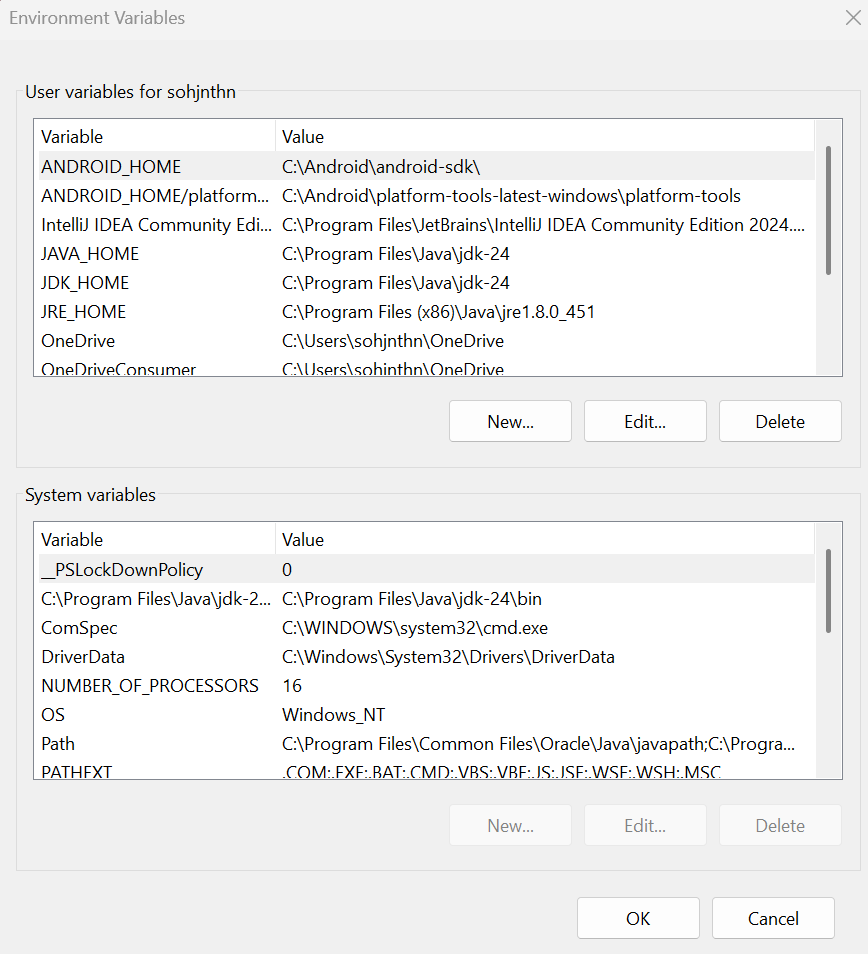
}

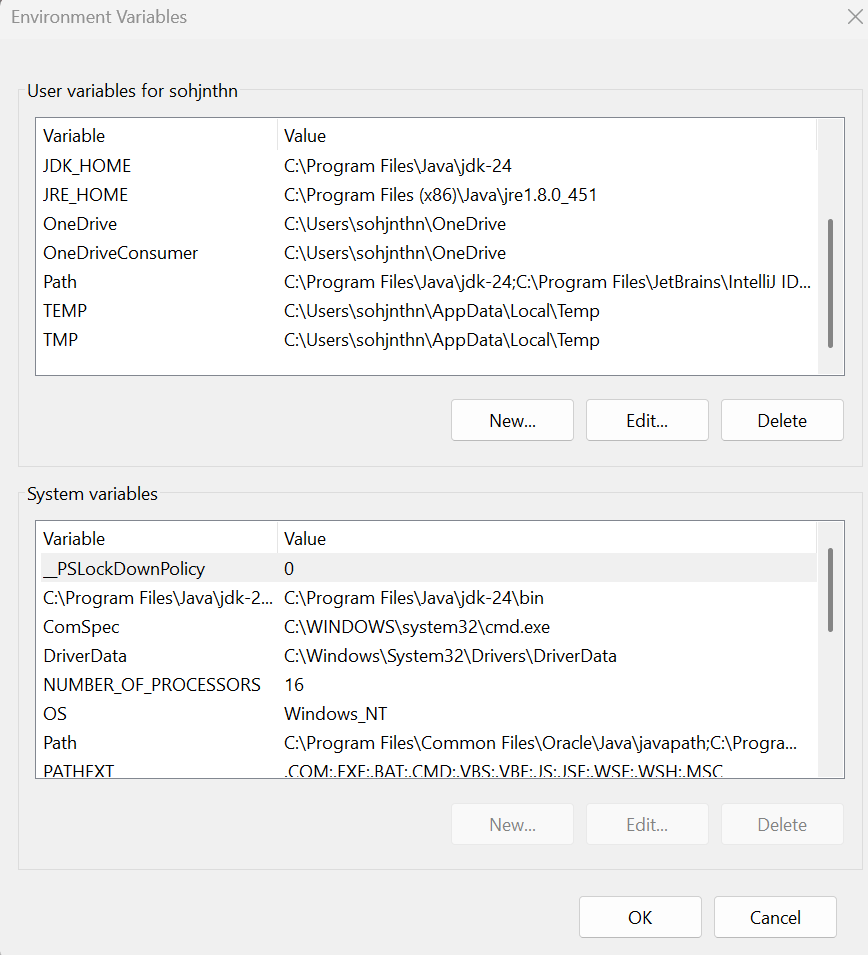
// There seems to be an error for which the two digits for day are incorrect, so localDateTime.getDayOfmonth is used(), instead of dayFirstDigitChar and daySecondDigitChar

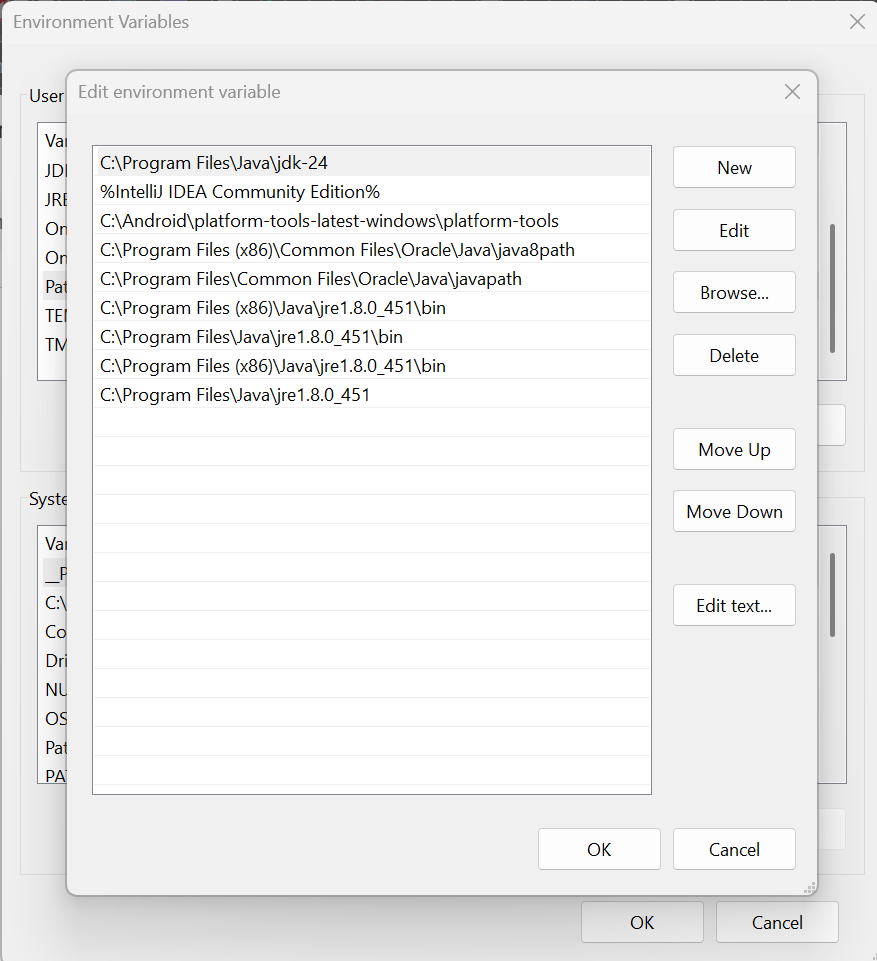
return localDateTime.getDayOfMonth() + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar;

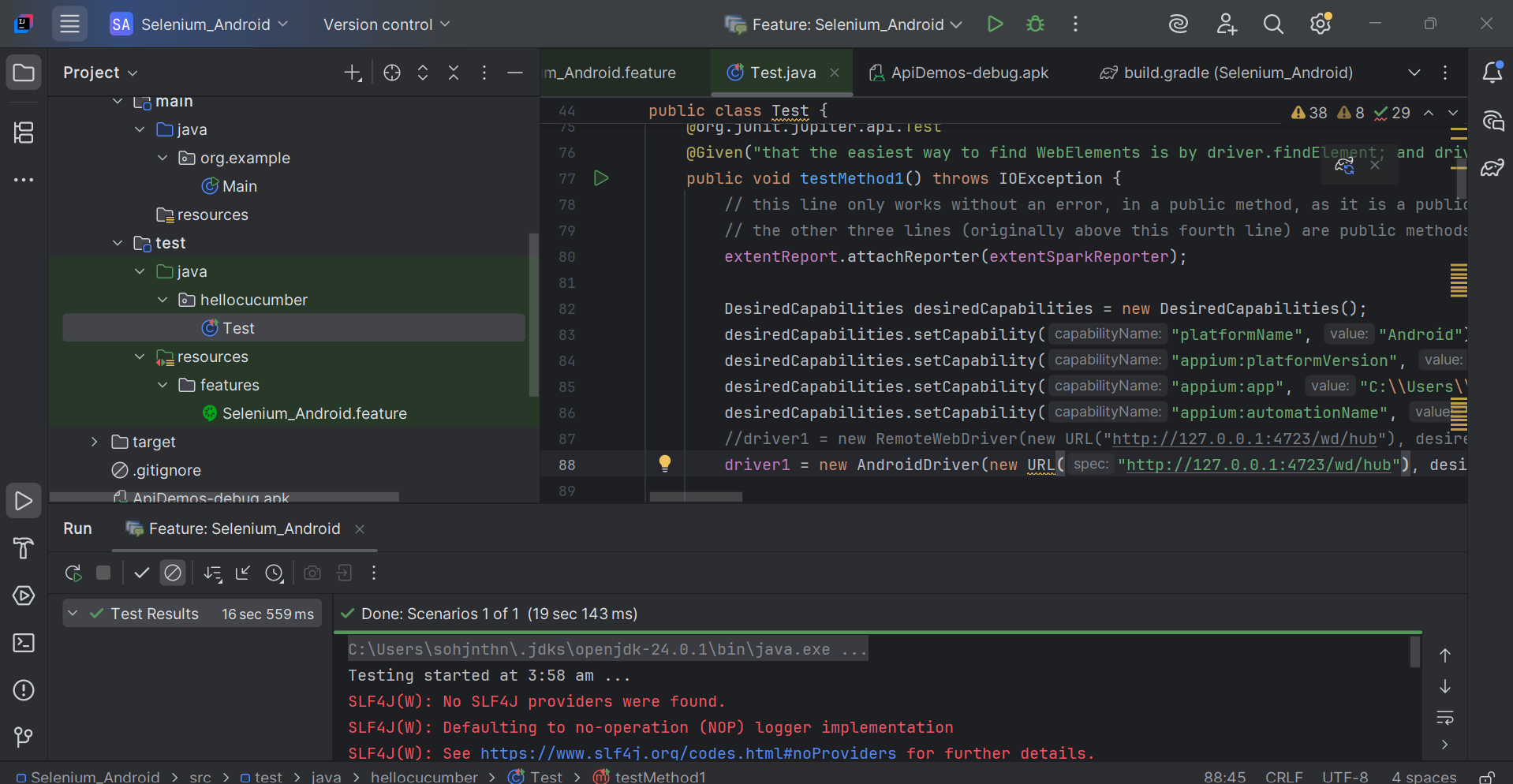
}

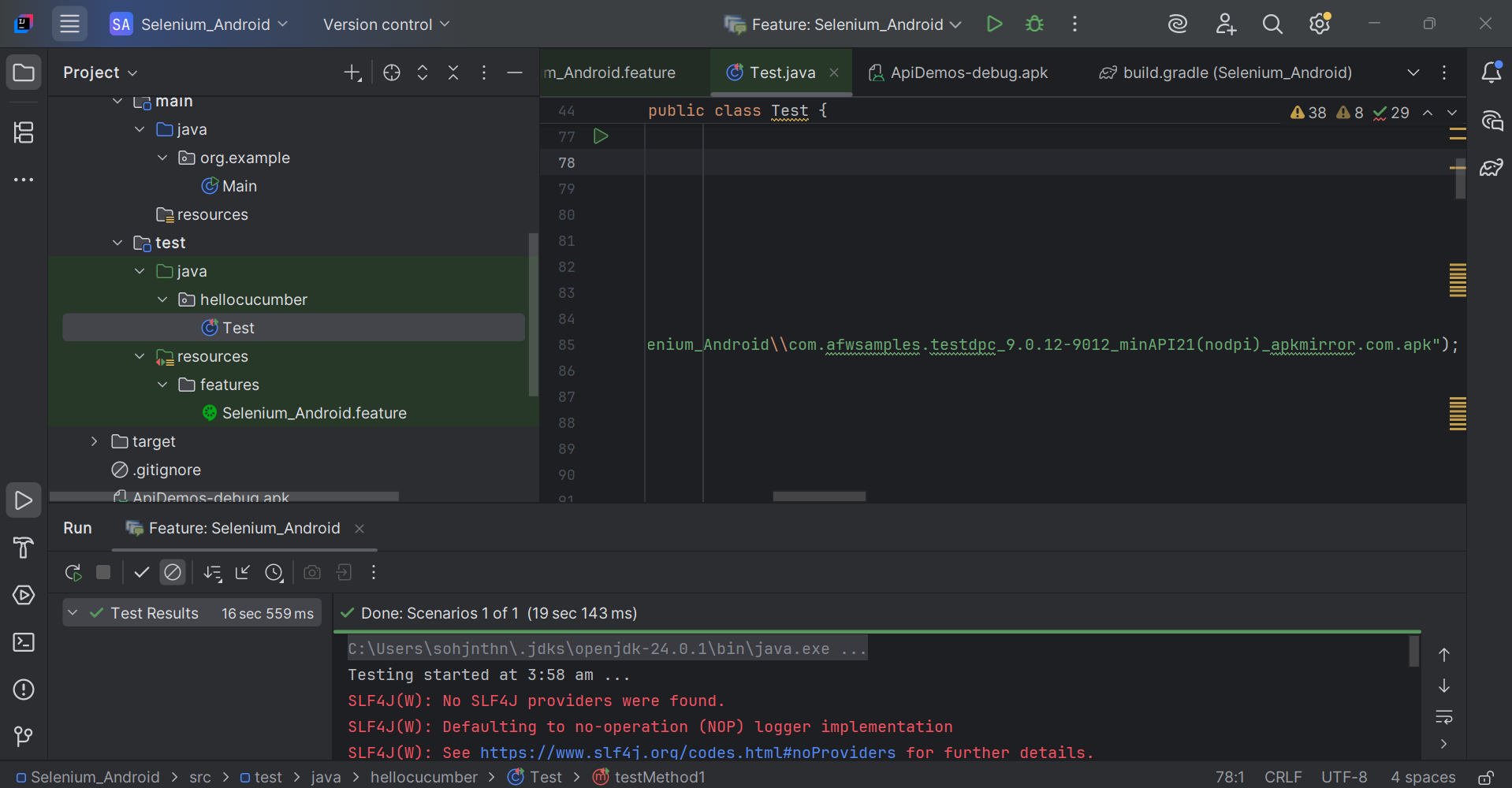
}

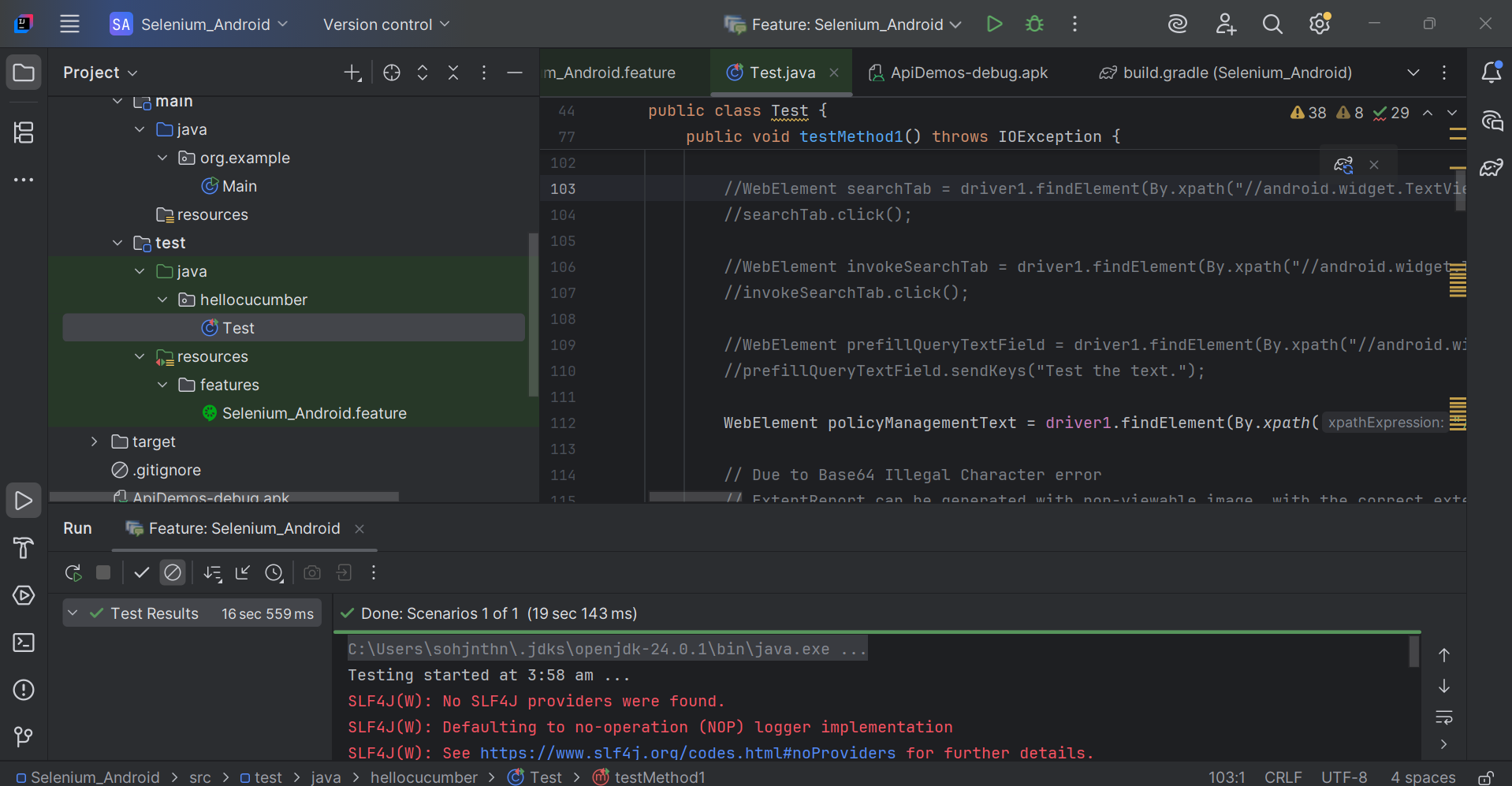


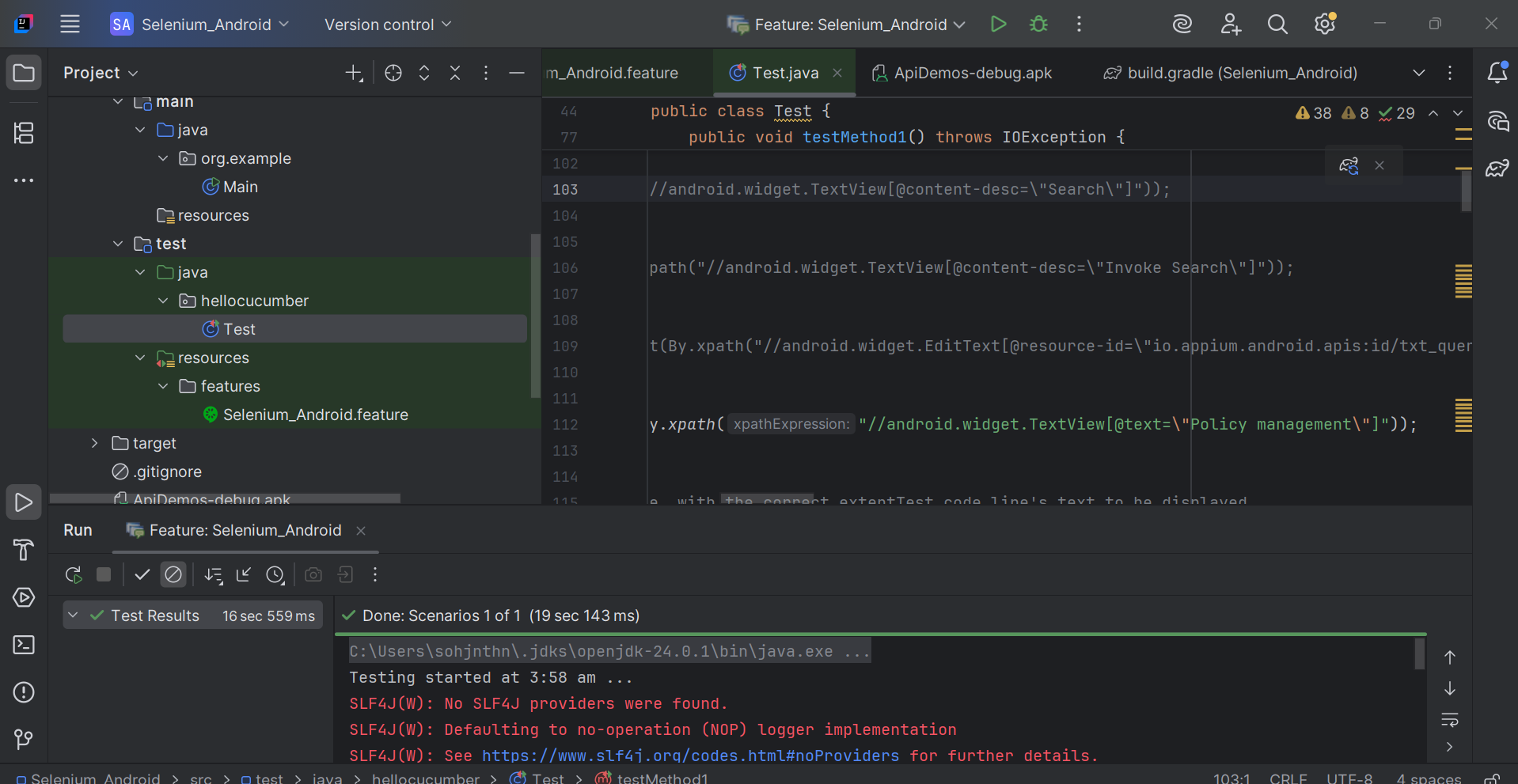


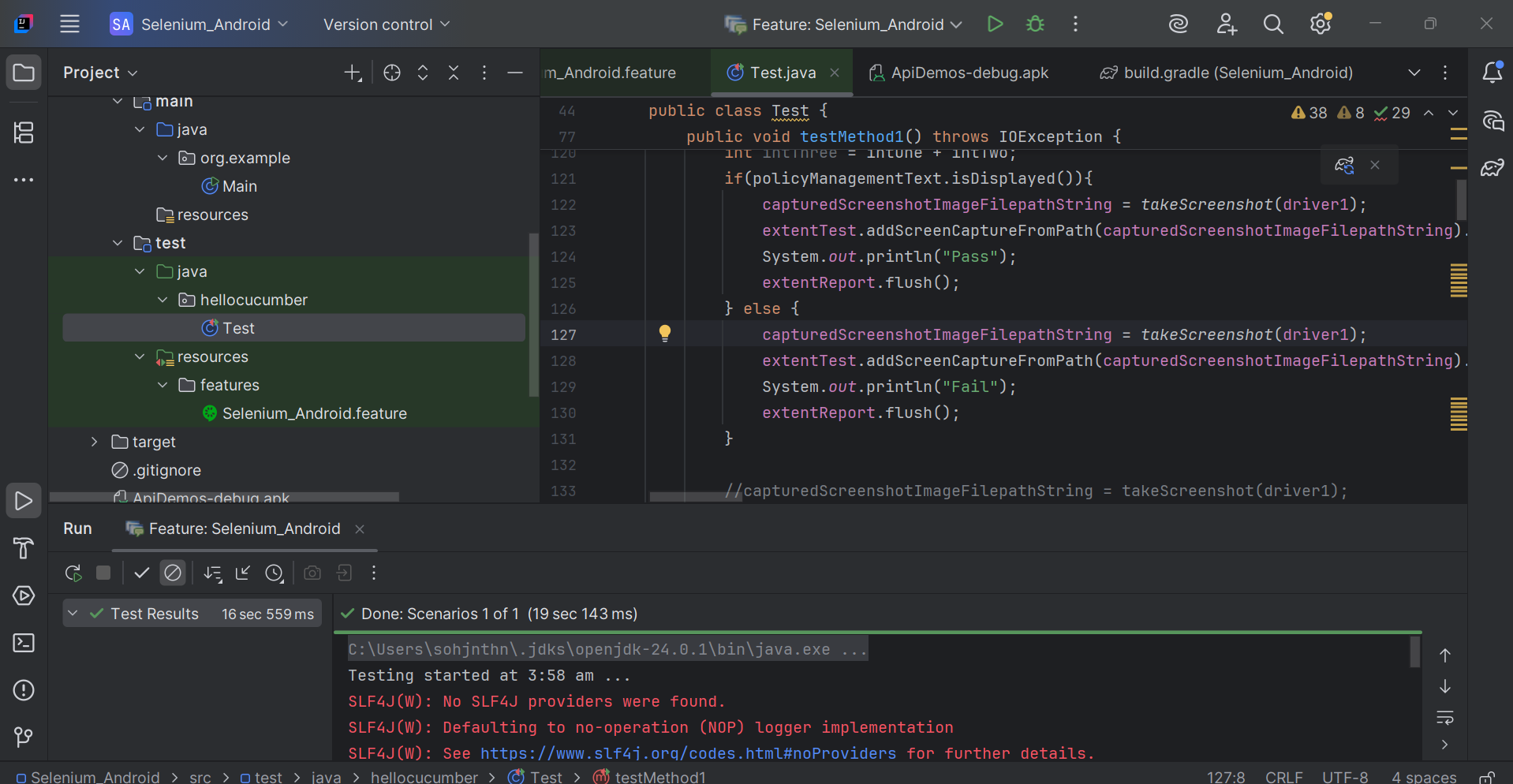


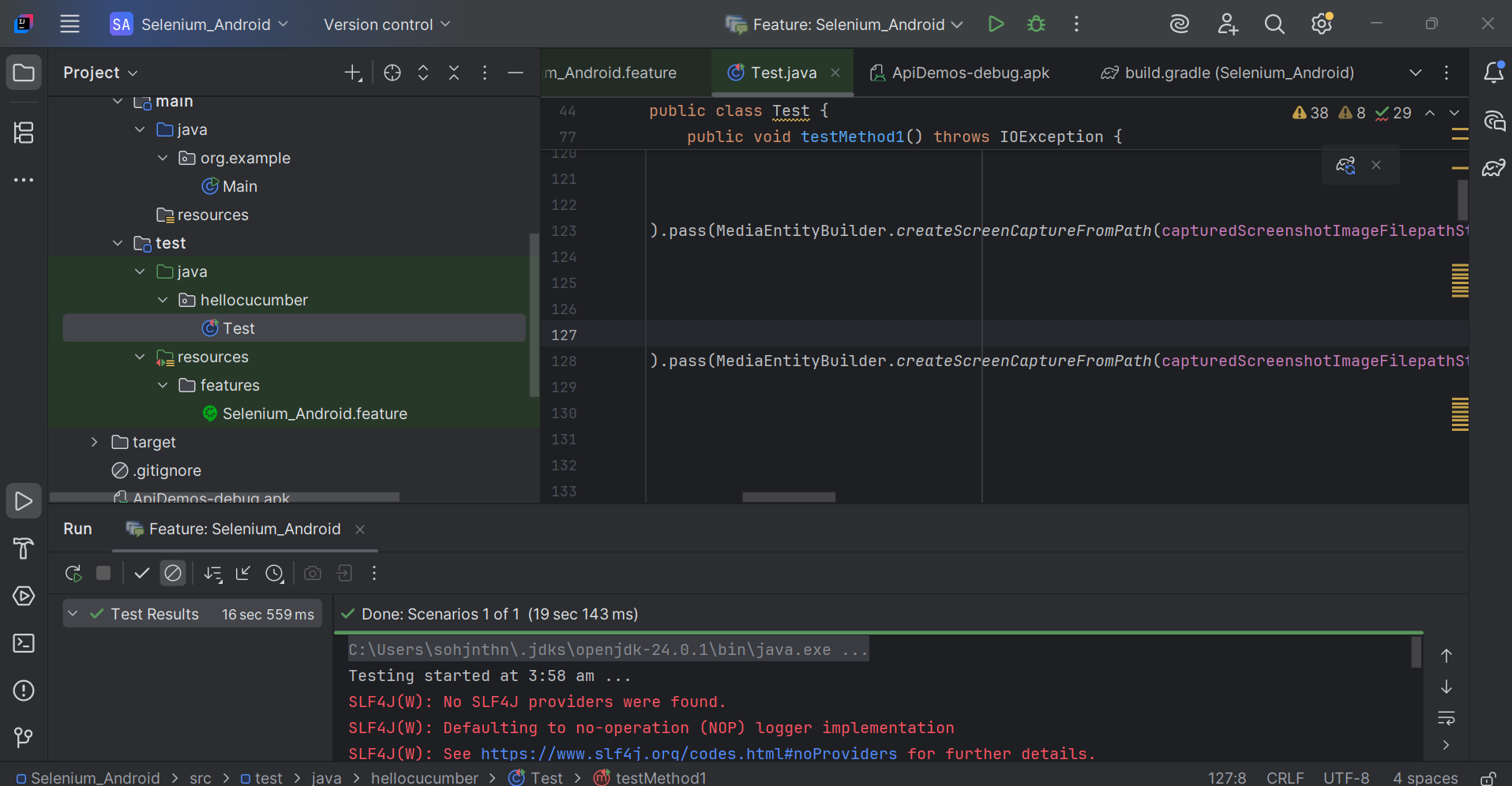




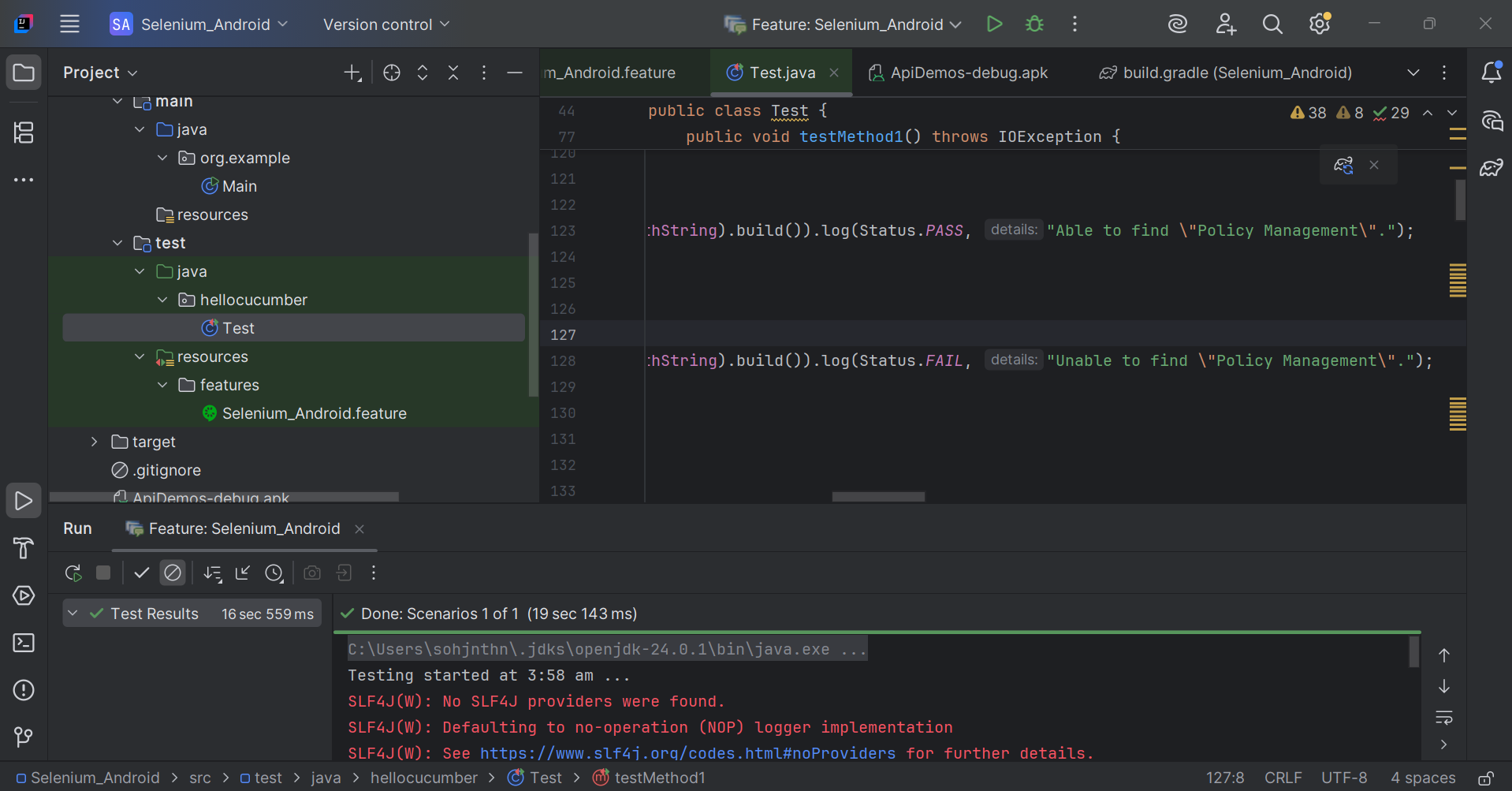


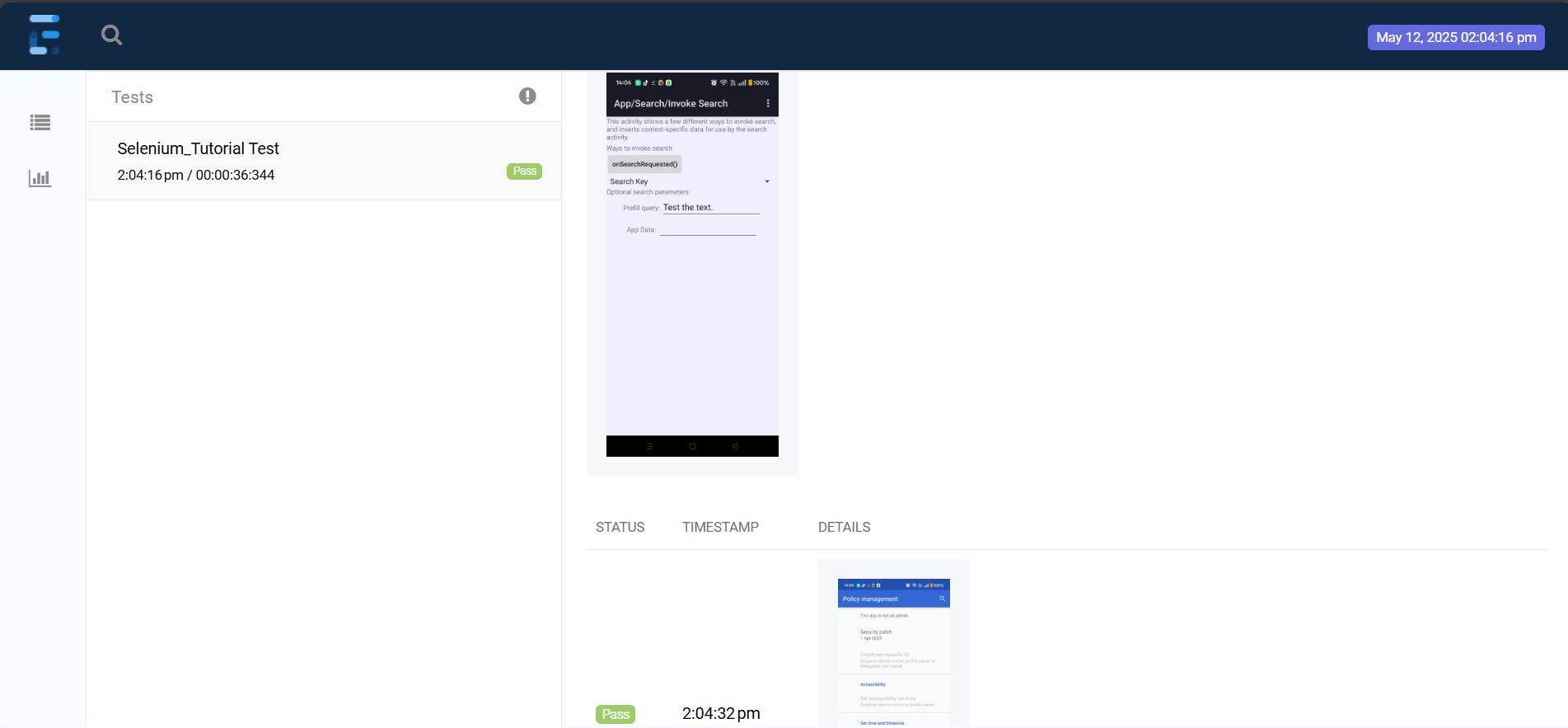


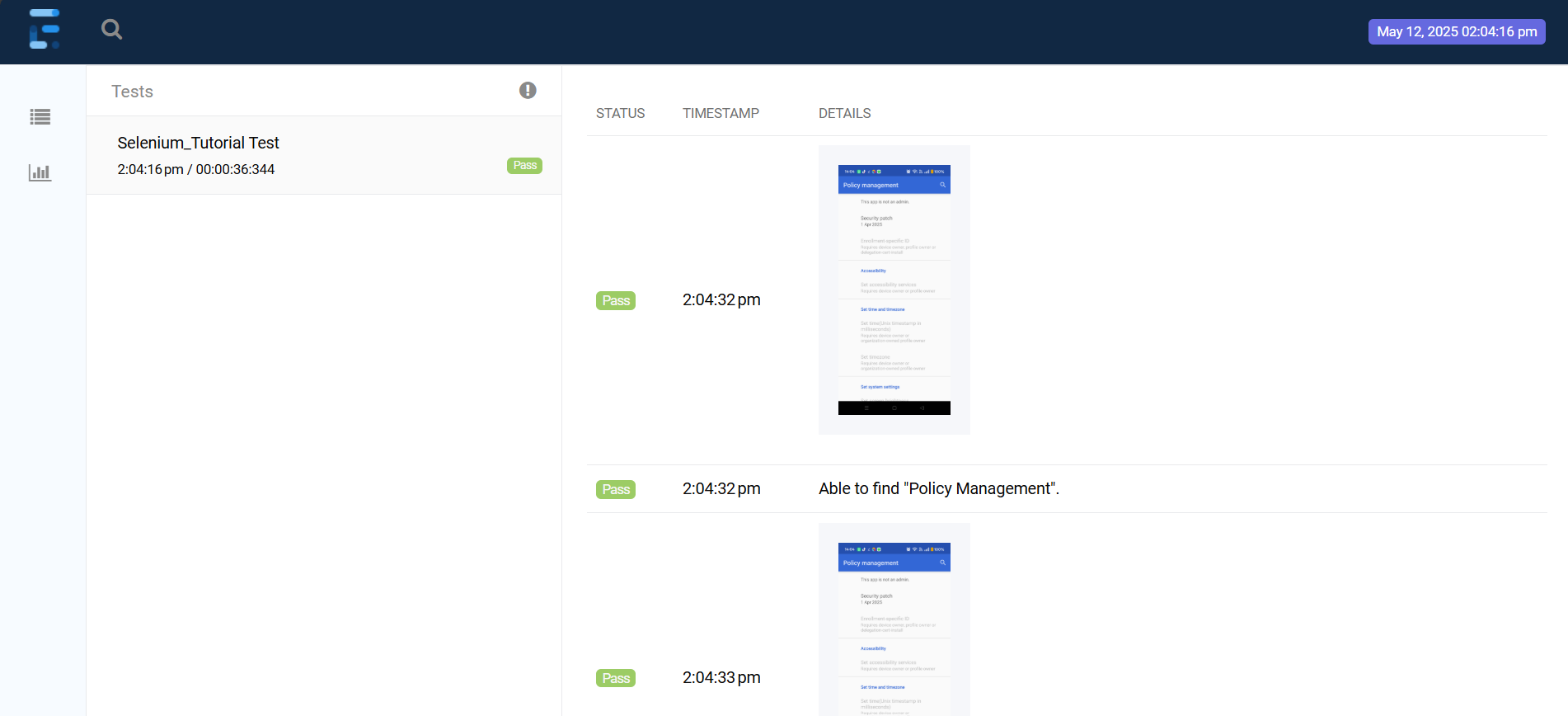


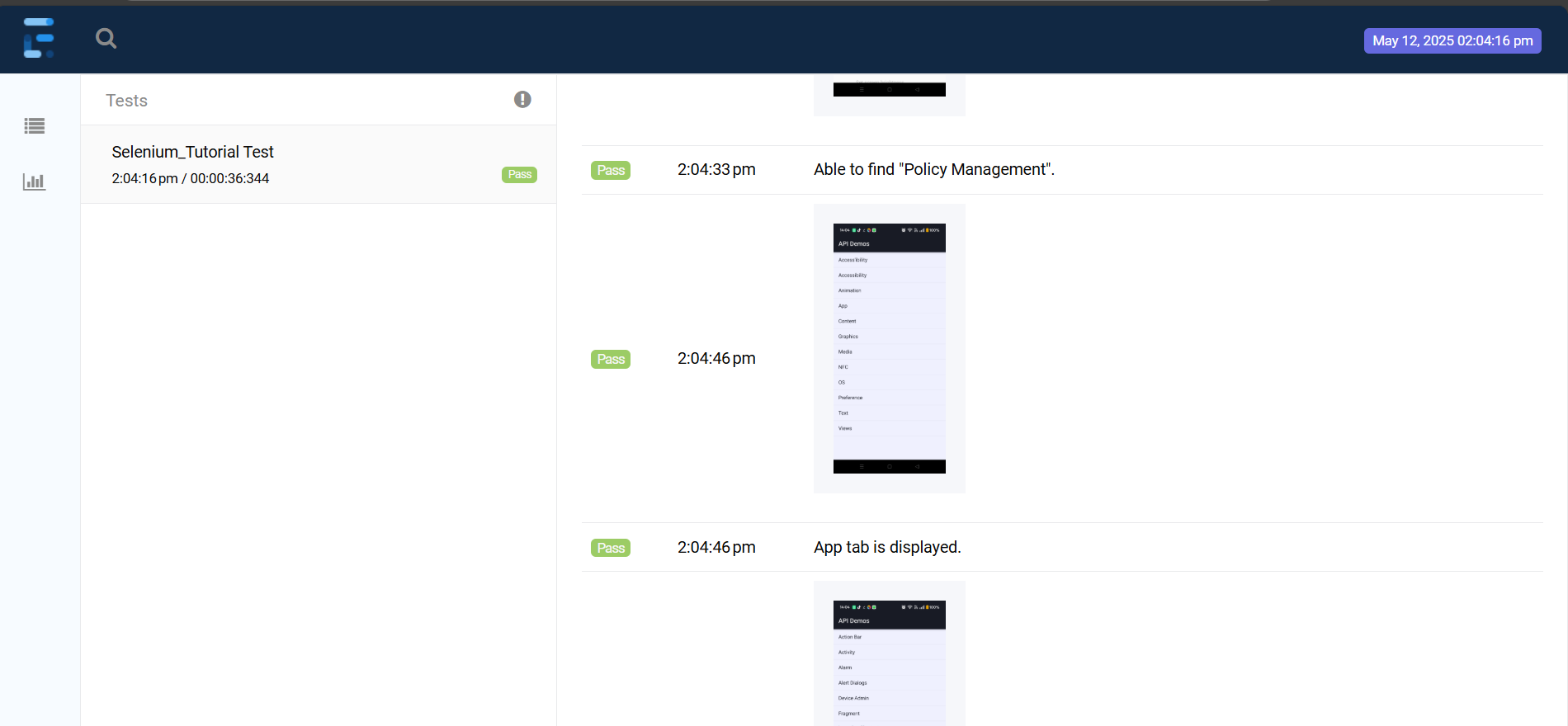


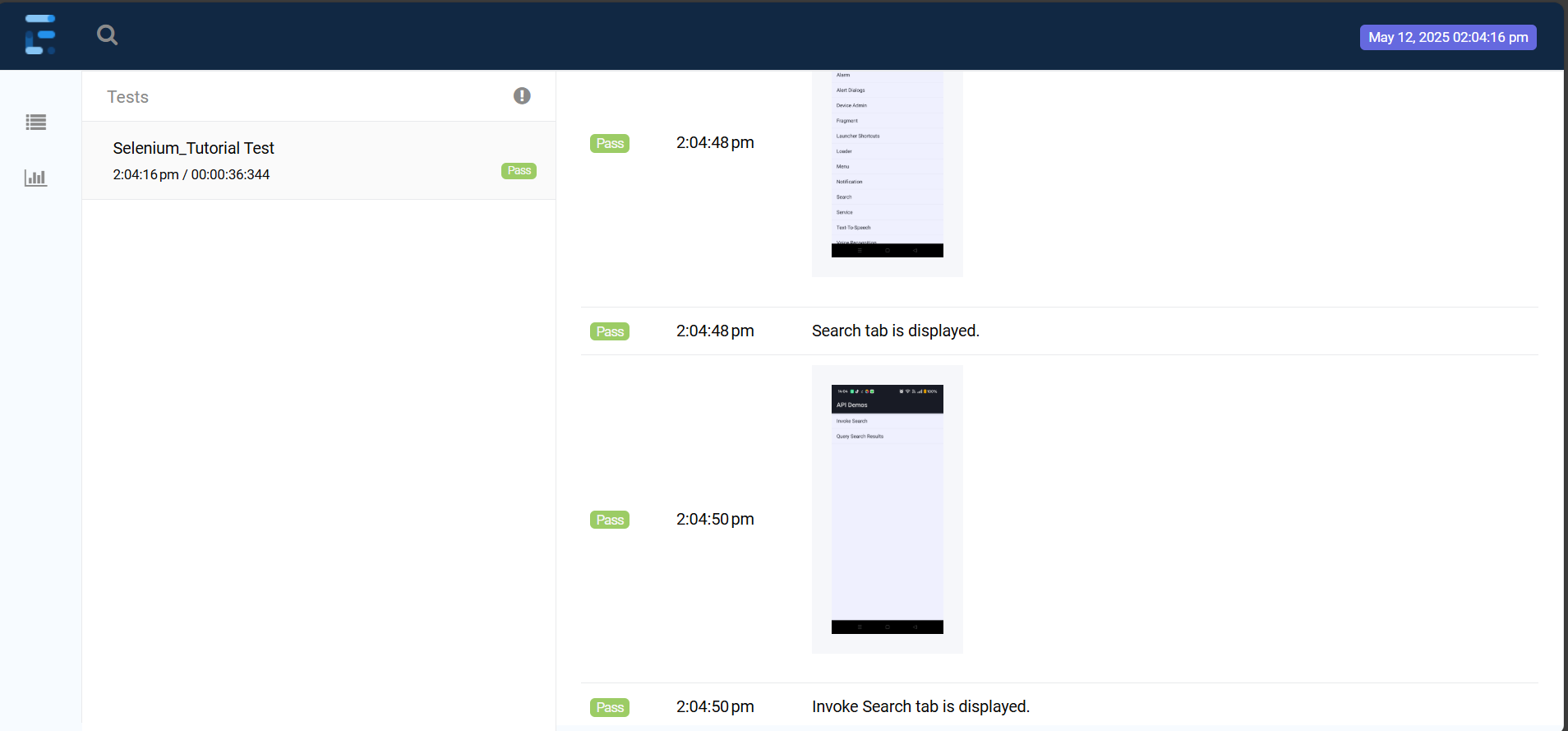
findElement entirely by xpath

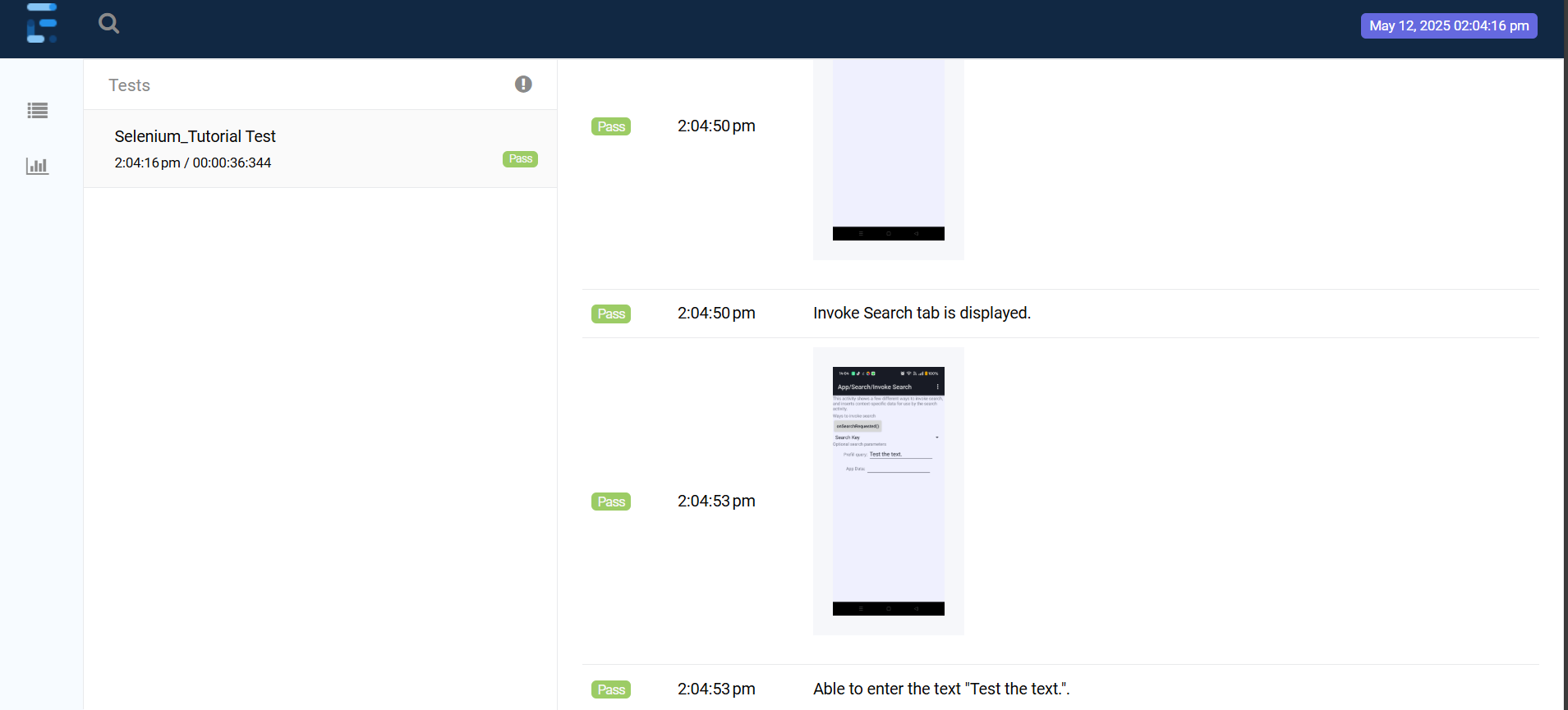


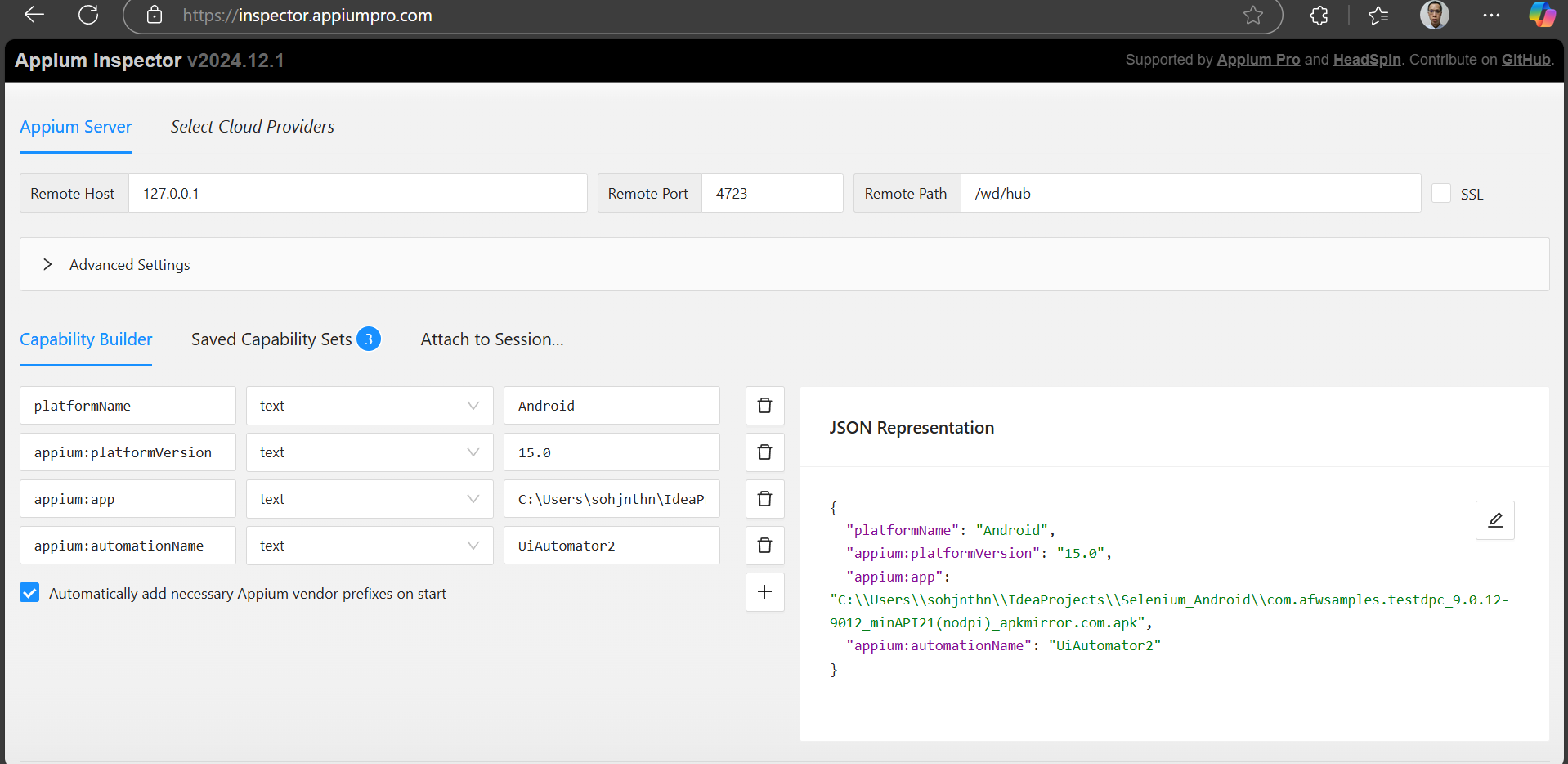


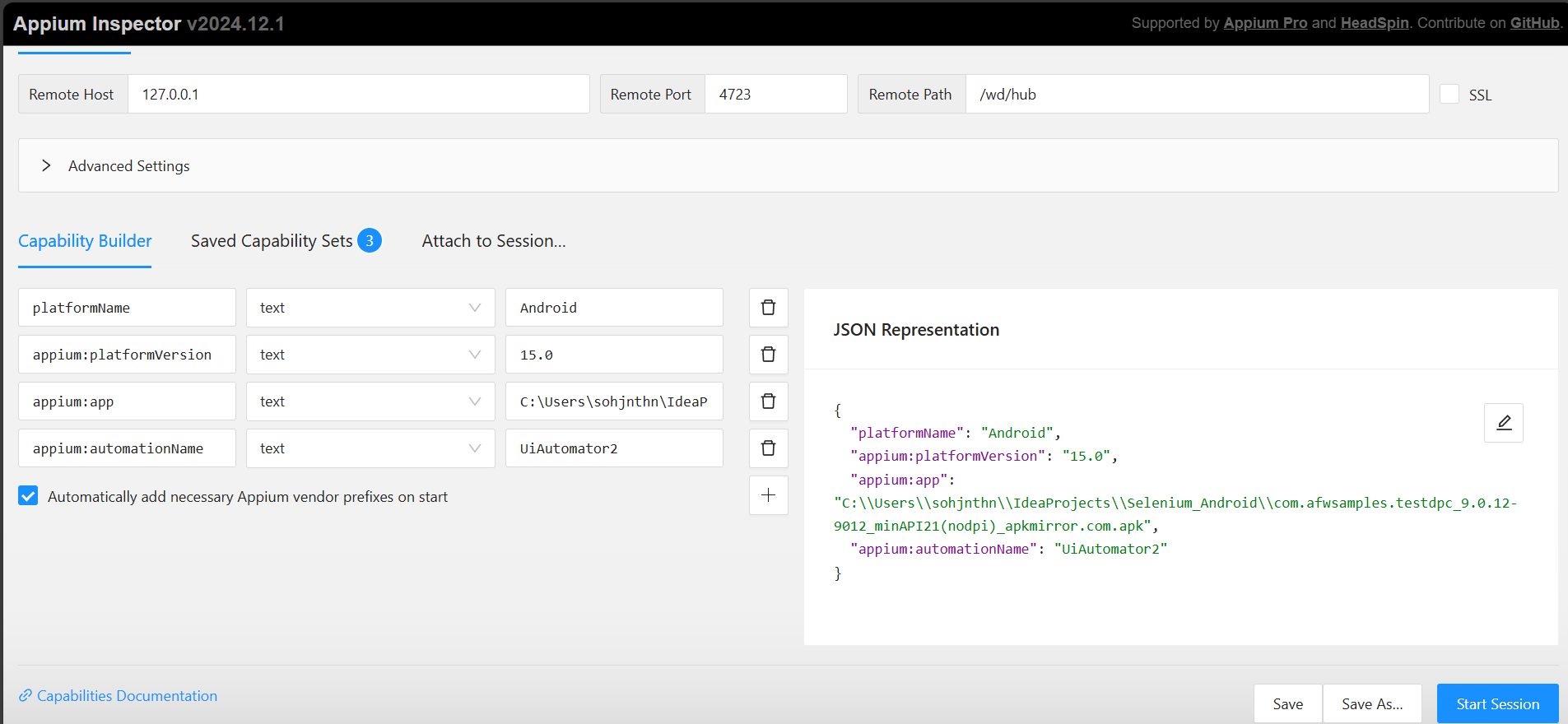


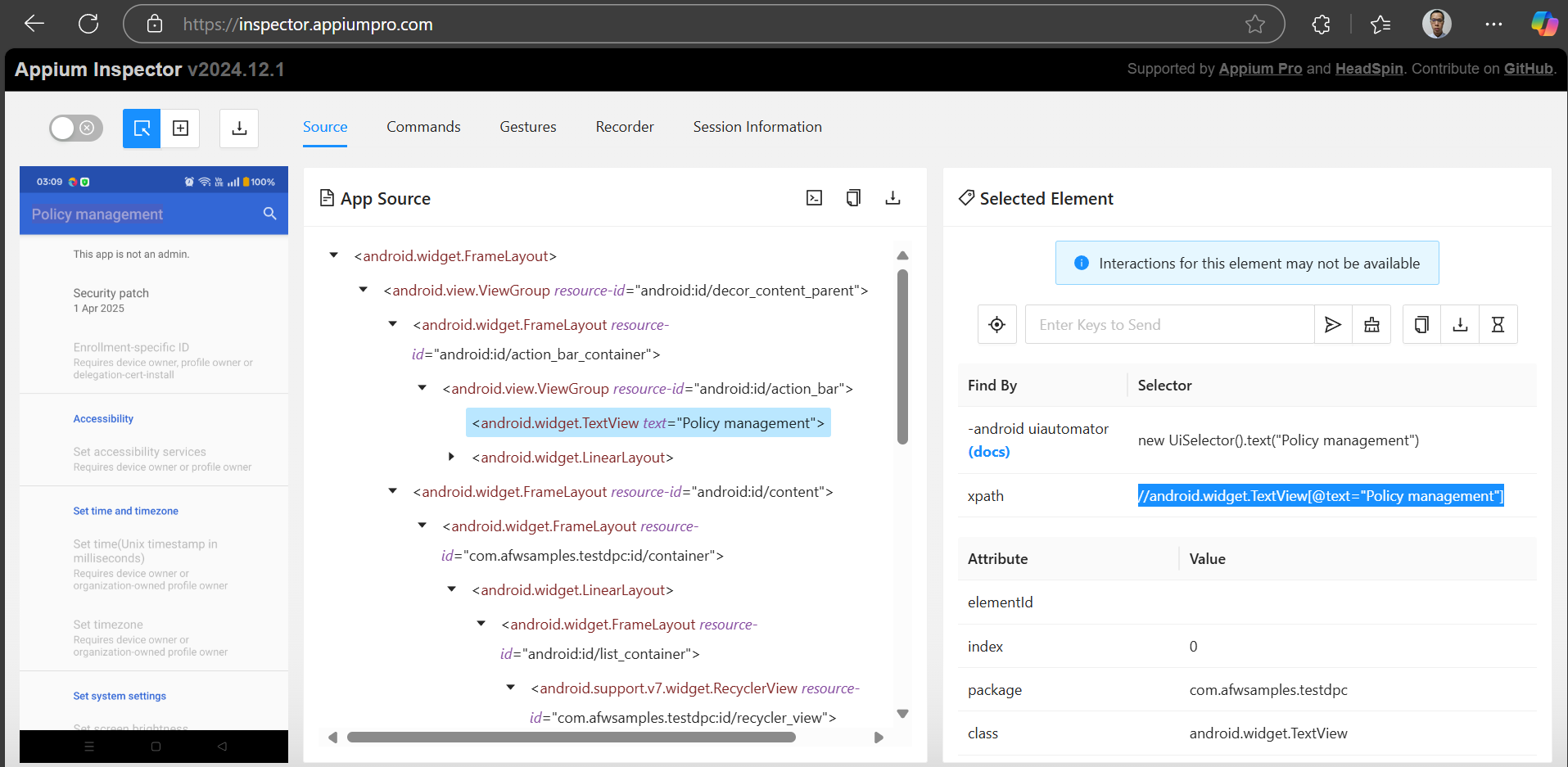




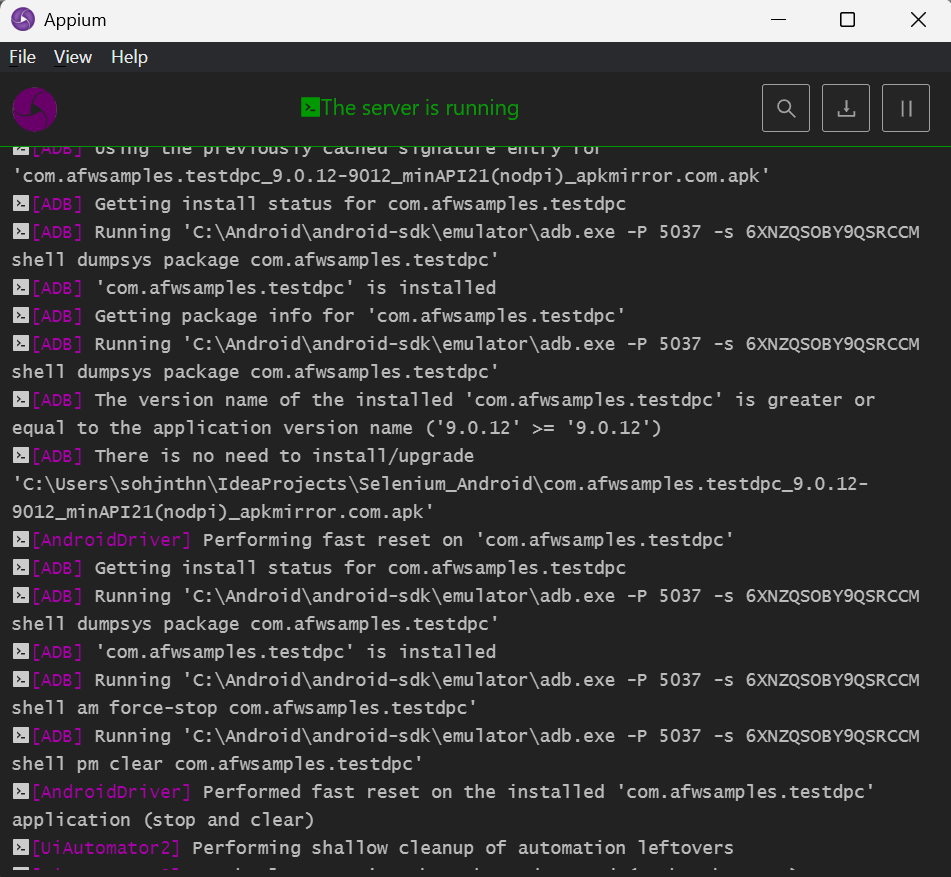








Remember to Allow CORS



build.gradle

plugins {

id 'java'

}

group = 'org.example'

version = '1.0-SNAPSHOT'

repositories {

mavenCentral()

maven { url = uri("https://jitpack.io")}

google()

gradlePluginPortal()

mavenLocal()

flatDir {

dirs "libs"

}

}

dependencies {

implementation 'io.appium:java-client:9.4.0'

implementation 'io.cucumber:cucumber-junit:7.22.1'

implementation 'org.junit.jupiter:junit-jupiter-api:5.12.2'

implementation 'org.junit.jupiter:junit-jupiter-engine:5.12.2'

implementation 'io.cucumber:cucumber-core:7.22.1'

implementation 'io.cucumber:cucumber-spring:7.22.1'

implementation 'org.apache.groovy:groovy-all:4.0.26'

implementation 'io.cucumber:cucumber-gherkin:7.22.1'

implementation 'io.cucumber:gherkin:32.1.1'

implementation 'io.cucumber:gherkin-utils:9.2.0'

implementation 'io.cucumber:cucumber-jvm-groovy:6.10.4'

implementation 'io.cucumber:cucumber-groovy:6.10.4'

implementation 'io.cucumber:cucumber-java:7.22.1'

implementation 'org.seleniumhq.selenium:selenium-chrome-driver:4.32.0'

implementation 'com.aventstack:extentreports:5.1.2';

implementation 'org.springframework:spring-beans:6.2.6'

implementation 'org.springframework:spring-core:6.2.6'

implementation 'org.springframework:spring-web:6.2.6'

implementation 'org.springframework:spring-test:6.2.6'

implementation 'org.springframework:spring-webmvc:6.2.6'

implementation 'org.springframework:spring-jms:6.2.6'

implementation 'org.springframework:spring-messaging:6.2.6'

implementation 'org.springframework:spring-aop:6.2.6'

implementation 'org.springframework:spring-orm:6.2.6'

implementation 'org.springframework:spring-context-support:6.2.6'

implementation 'org.springframework:spring-context:6.2.6'

implementation 'org.springframework:spring-tx:6.2.6'

implementation 'org.springframework:spring-jdbc:6.2.6'

}

test {

useJUnitPlatform()

}