

**DIGITAL ASSIGNMENT - II**

**COURSE NAME:** ADVANCED SOFTWARE TESTING

**COURSE CODE:** SWE3006

**FACULTY NAME:** PROF. NACHIYAPPAN.S

**TEAM MEMBERS:**

1. BLESSY BOBAN - 17MIS1014
2. PREETHA SARAVANAN – 17MIS1069

**ABSTRACT:**

In classical café, restaurants and hotels, the customers face a lot of problems due to congestion at peak hours, unavailability of waiters and due to manual order processing. Orders of customers can be taken in seconds. The kitchen receive the orders as soon as the waiters take them. The app is recommended for any waiter and cook who doesn't want to fumble around with pen and paper. This order is sent to the kitchen and reception using communication network. Then the waiter transfers the food from the kitchen to the customer.

**► TAKE ORDERS FASTER**  
A couple of taps and there you go, the customer's order was taken and sent to the kitchen.  
  
**► CREATE THE BEST MENU**  
Customize your menu with categories and extra toppings and share it with all the rest of the staff.  
  
**► INVITE YOUR FELLOW CO-WORKERS**  
Invite your colleagues to use Waiterio so orders and menu are automatically shared.  
  
**► RECEIVE PAYMENTS**  
Waiterio is a fully functional POS system. Calculate the total for an order and receive the payments by cash or credit card.

**TEST ENVIRONMENT:**

COMPONENTS IN SOFTWARE TEST ENVIRONMENT:

* Android7
* Java 8.1
* Windows10

SOFTWARE ITEMS UNDER TEST:

* Experitest
* Android studio
* Expertise(see test)

**PERFORMANCE TESTING:**

NON-FUNCTIONAL:

* Create an bulid application as apk .
* Sign up to experitest testing tool which in an open source for 24 hours.
* Add the apk file of the created application to test the performance.
* Select the device in which we are going to run our application.
* Click reporting and then start recording, do Some functions on that application that are to be tested.

FEATURES:

* note for item
* extra topping for item
* courses
* split bill
* service charge
* tax
* print daily report
* export report to .csv
* login with PIN
* print using thermal printers

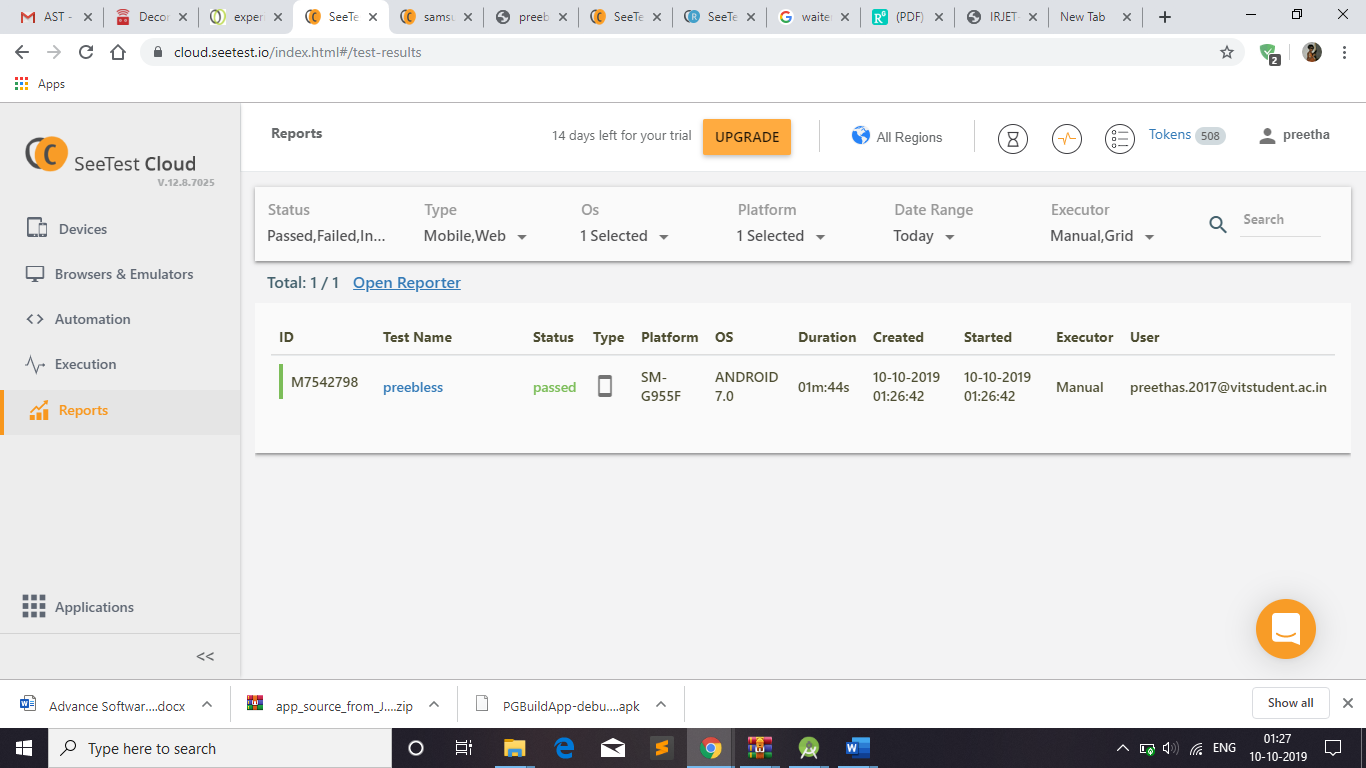
WORKS AS:

* Restaurant POS System
* Restaurant Point of Sale Free
* Restaurant Software
* Ordering system
* Table management system
* POS Bar
* Point of Sale Bar
* Coffee shop POS
* Coffee shop Point of Sale
* Cafe Point of Sale'

PROCEDURES:

* Upto 5 min we can record.
* Stop recording, save the report.
* Automation button in left panel, select the OS and category of the application.
* Os version and app type will be visible.
* Framework and language will be in the right Panel
* After running the script queued test,running test will be done
* Test result will be generated as pdf
* View the reports and take screenshots.
* Open the first tab->reports->test case status
* Take the screenshots of the test case reports(pass/fail).

**TEST REPORTS:**



**TESTING TYPES USED:**

* Performance Testing
* Functional Testing

**PERFORMANCE TESTING:**

We had Used (see test) which is hosted by expertise to test following feature

* Duration
* Speed Index
* CPU
* Memory
* Battery
* Network

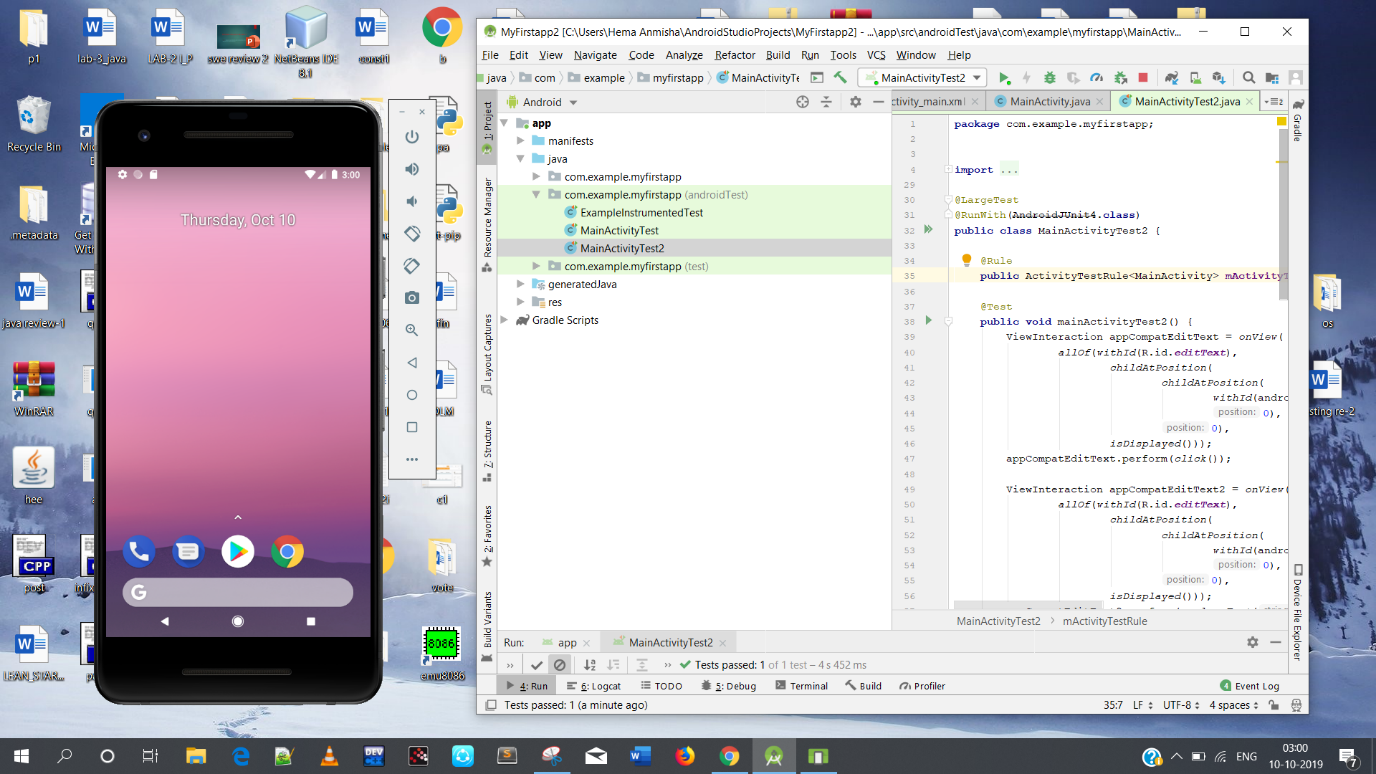
**TEST CASE:**

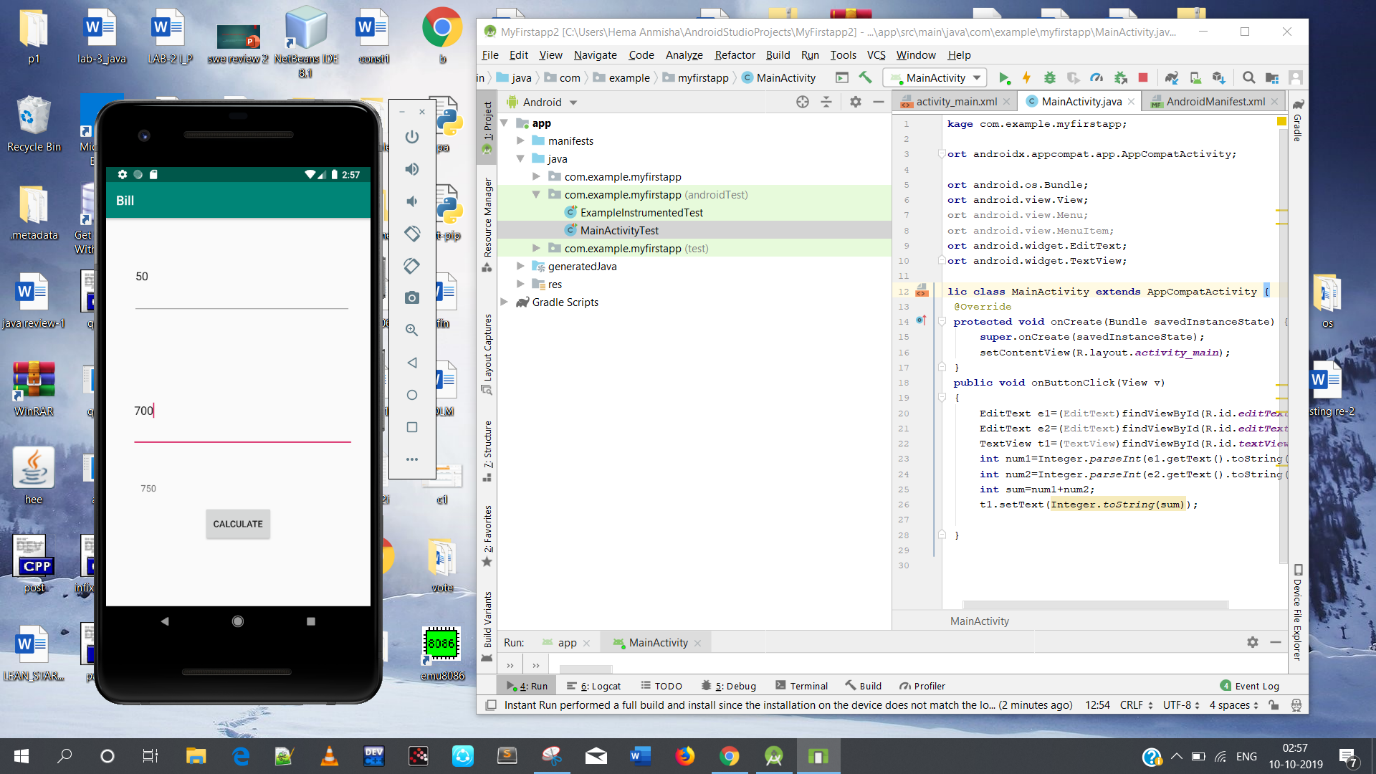
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TEST CASE ID | TEST SCENARIO | TEST DATA | TEST STEPS | EXPECTED  OUTPUT | ACTUAL OUTPUT | PASS / FAIL |
| TC1 | On click | Open the application | Refresh screen | Display the screen | Display the screen | pass |
| TC2 | Selecting tables | Number of tables available | Onclick the vacancy tables | Selects only the vacancy tables | Selects only the vacancy tables | pass |
| TC3 | Adding the orders | Item names | Onclick the ordering items | Selecting the items | Selecting the items | pass |
| TC4 | Calculating the amount | Cost of the items | View the amount | Shows the amount | Show the amount | pass |
| TC5 | Payment button | Card details | Onclick | Successfully paid | Successfully paid | pass |

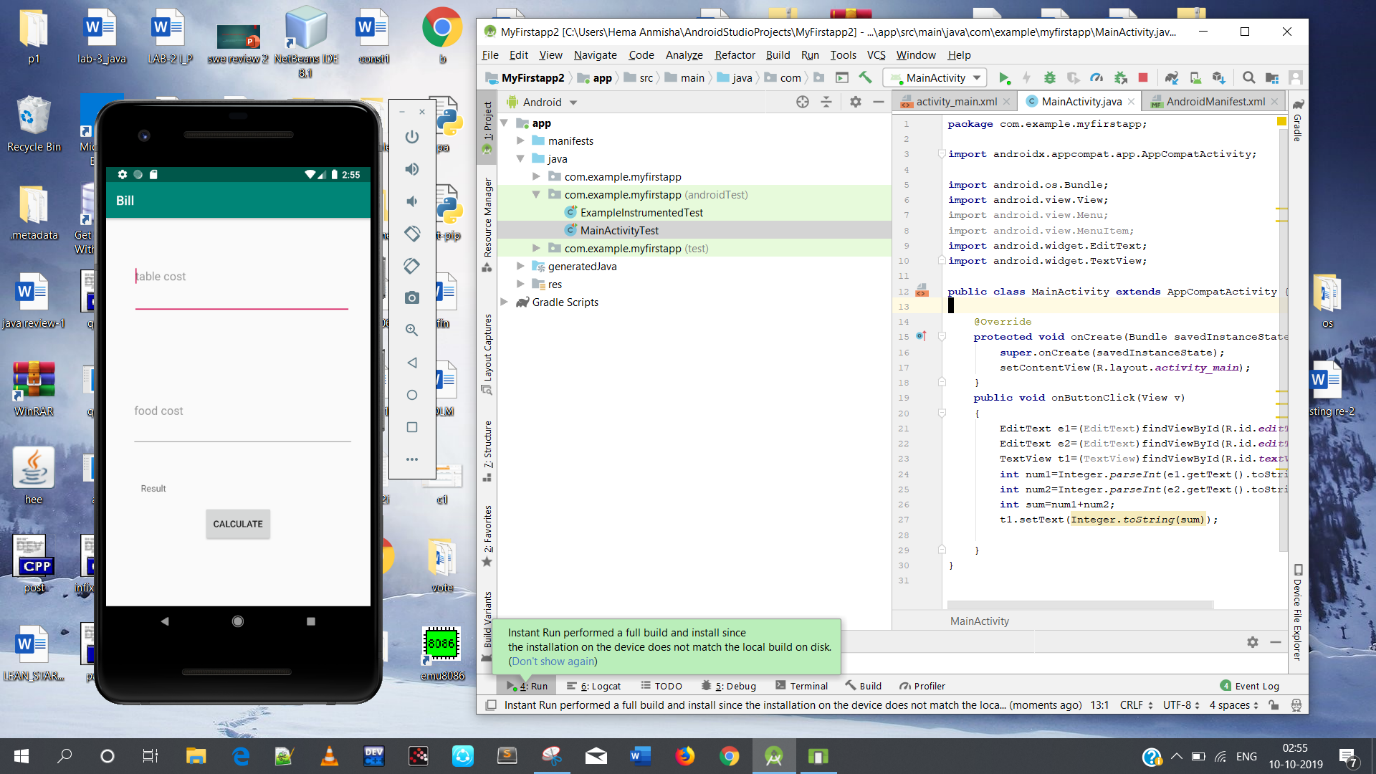
**FUNCTIONAL TESTING:**

**PROCEDURE:**

* Extract the given app in Android Studio (Profile/Debug) and convert all the small file types to the java Dex
* Perform the Gradle sync in the Studio for the dependencies of the files and Prepare for the Resource monitor of the App.
* Run the Given app from the AVD manager and select virtual or physical device.
* Once the run starts perform some tasks that are intensive to the app to observe the graph battery graph and CPU usage
* From the palette window.drag and drop two number fields and position them according to your choice on the layout
* Double click on each of the icon and remove the text in both the number fields
* Now drag and drop a button and change the text to Pay
* Save your program and run it in your emulator to see the functioning of the app
* Run->record as expression
* Now perform the necessary actions for your applications
* The test scripts and generated
* Add assertions and save them and run the whole process
* And the status of the test will be shown.







**NON – FUNCTIONAL:**

**TEST SCRIPT:**

import io.appium.java\_client.ios.IOSDriver;

import io.appium.java\_client.ios.IOSElement;

import io.appium.java\_client.remote.IOSMobileCapabilityType;

import io.appium.java\_client.remote.MobileCapabilityType;

import org.junit.\*;

import org.openqa.selenium.By;

import org.openqa.selenium.ScreenOrientation;

import org.openqa.selenium.remote.DesiredCapabilities;

import java.net.MalformedURLException;

import java.net.URL;

public class LocaliOSTest {

private String accessKey = "eyJ4cC51Ijo3NTQyMTI1LCJ4cC5wIjo3NTQyMTI0LCJ4cC5tIjoiTVRVM01EWTBOalUxTWpNd05BIiwiYWxnIjoiSFMyNTYifQ.eyJleHAiOjE4ODYwMDY2MzAsImlzcyI6ImNvbS5leHBlcml0ZXN0In0.IVY\_1OK3hl2ZH1bTOW5lgZUcvMdOTzD2hQkMibGa82g";

protected IOSDriver<IOSElement> driver = null;

DesiredCapabilities dc = new DesiredCapabilities();

@Before

public void setUp() throws MalformedURLException {

dc.setCapability("testName", "Quick Start iOS Native Demo");

dc.setCapability("accessKey", accessKey);

dc.setCapability("deviceQuery", "@os='ios' and @category='PHONE'");

dc.setCapability(MobileCapabilityType.APP, "cloud:com.experitest.ExperiBank");

dc.setCapability(IOSMobileCapabilityType.BUNDLE\_ID, "com.experitest.ExperiBank");

driver = new IOSDriver<>(new URL("https://cloud.seetest.io/wd/hub"), dc);

}

@Test

public void quickStartiOSNativeDemo() {

driver.rotate(ScreenOrientation.PORTRAIT);

driver.findElement(By.xpath("//\*[@id='usernameTextField']")).sendKeys("company");

driver.hideKeyboard();

driver.findElement(By.xpath("//\*[@id='passwordTextField']")).sendKeys("company");

driver.findElement(By.xpath("//\*[@id='loginButton']")).click();

driver.findElement(By.xpath("//\*[@id='makePaymentButton']")).click();

driver.findElement(By.xpath("//\*[@id='phoneTextField']")).sendKeys("0541234567");

driver.findElement(By.xpath("//\*[@id='nameTextField']")).sendKeys("Jon Snow");

driver.findElement(By.xpath("//\*[@id='amountTextField']")).sendKeys("50");

driver.findElement(By.xpath("//\*[@id='countryButton']")).click();

driver.findElement(By.xpath("//\*[@id='Switzerland']")).click();

driver.findElement(By.xpath("//\*[@id='sendPaymentButton']")).click();

driver.findElement(By.xpath("//\*[@id='Yes']")).click();

}

@After

public void tearDown() {

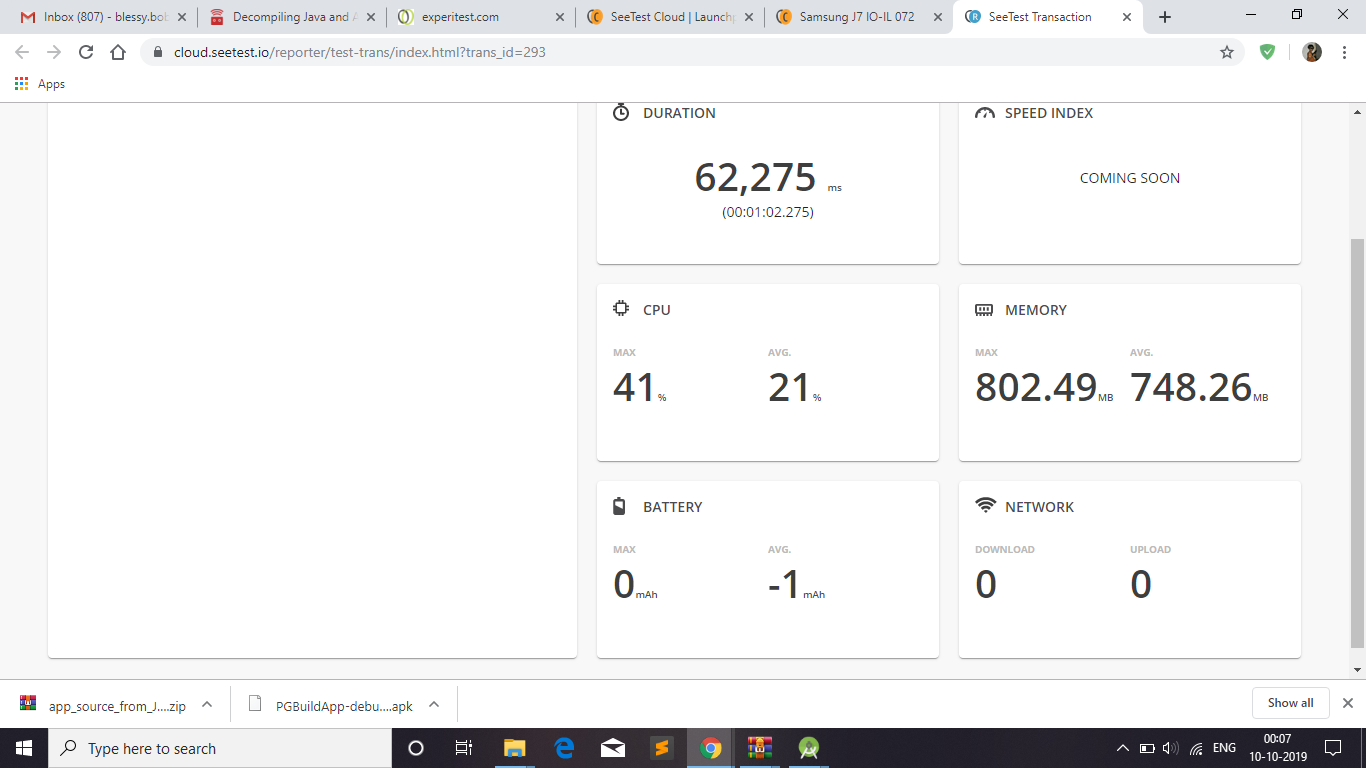
System.out.println("Report URL: "+ driver.getCapabilities().getCapability("reportUrl"));

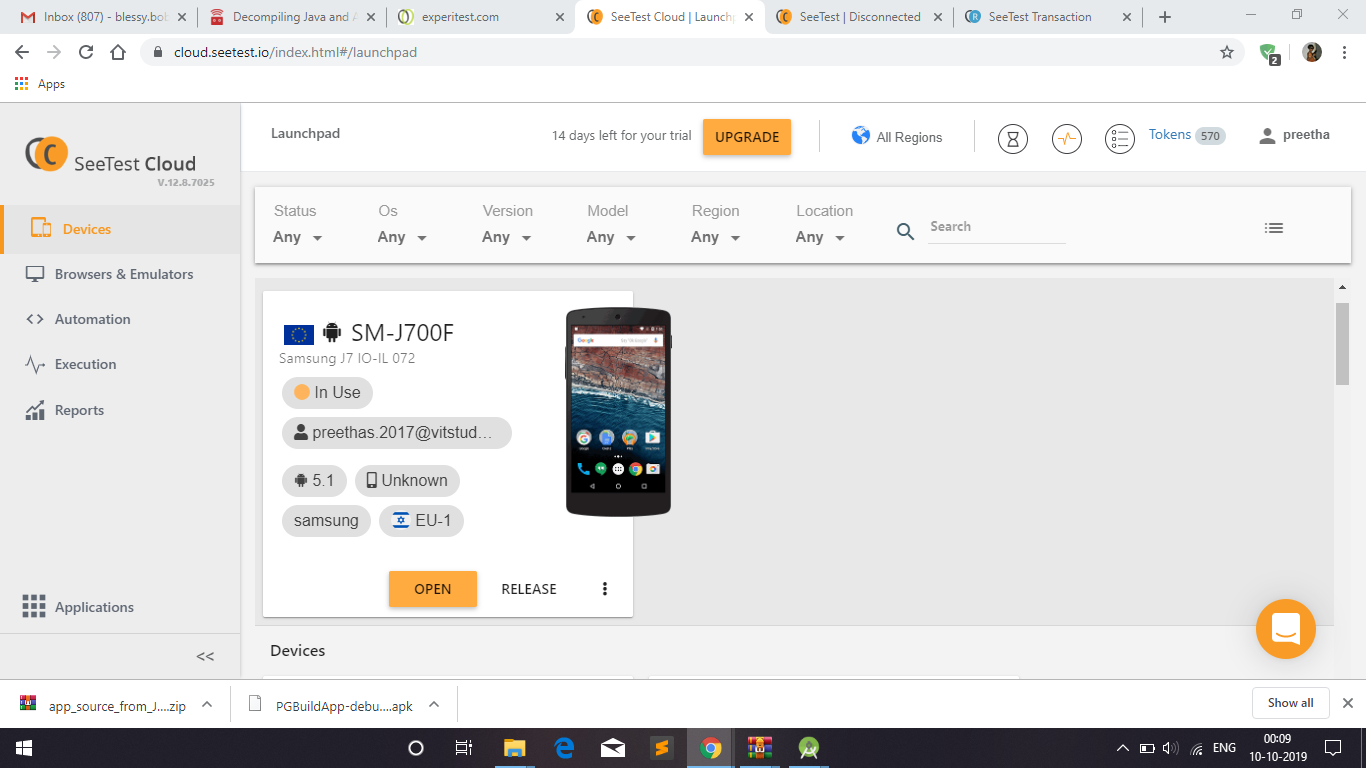
driver.quit();

}

}

SCREENSHOTS:





**APPLICATION SCREENSHOTS:**



