**Appium Notes**

**Appium**: Open source Mobile Automation tool which is Internally work as similar to Selenium WebDriver (Json over HTTP) and support all the API's which Selenium support + support Appium libraries.

Support Cross platform: IOS, Android

Support multiple language: Java, JS, Python, Ruby, C#

Support Apps: Native Apps, Web Apps and Hybrid Apps

**Apk file** - apk file is just like exe file in windows, which are executable in mobile.

adb : android device bridge

apk: Android application package

**Installing Appium with Node.js**

Step 1: Check if node.js is installed on your system

node --version npm --version

Step 2: Download node.js installer

https://nodejs.org/en/download/

Step 3: Run the installer & install node.js & npm

Step 4: Check if node.js & npm are installed

node --version

npm --version

where node

where npm

Step 5: Install Appium with node.js

Appium comes as node module, that is why npm is necessary and we can download any node module using npm.

npm install -g Appium (g stands globally)

npm uninstall -g appium (This is used to uninstall appium)

Step 6: Check if appium is installed

appium -v or appium --version

where appium

Step 7: Start appium : type “appium” on cmd

Stop appium : Control C - for appium termination

**Default port for Appium: 4723**

**==============================================================================**

Step 1: Download Android Studio

https://developer.android.com/studio

comes as a two folder: one is SDK and another is Studio

Step 2: Set environment variables

ANDROID\_HOME = location of sdk folder

Path: add tools, add plateform-tools, add build-tools

Step 3: Check command adb devices on command line

Step 4: Make device ready

go to phone and **“On”** developer mode

Then go to additional settings - Developer options - on USB debugging - on USB debugging (Security settings) - connect to Vysor.

Step 5: Connect device to computer system through USB cable and if asked enable USB Debugging

Step 6: Run command **“adb devices”** and Check your device id displayed.

And if adb devices shows Unauthorized than 1st kill the server - start the server and check again “adb devices”

//adb devices- Unauthorized

//adb kill-server

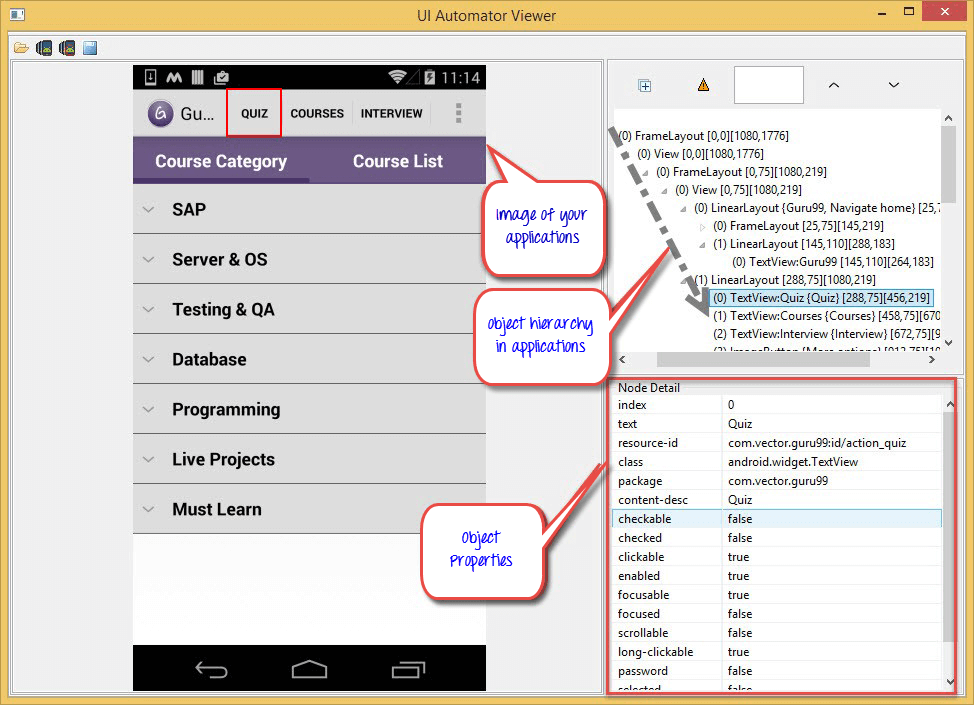
//adb start-server

//adb devices

**UiAutomatorViewer :**

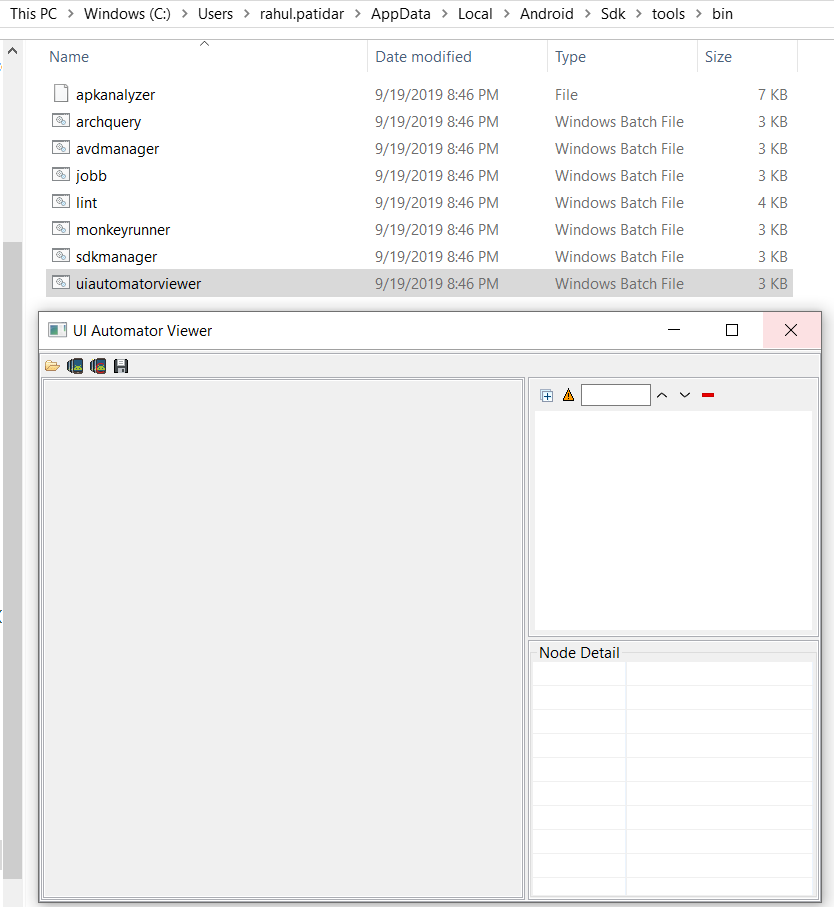
"UIautomatorviewer" is a GUI tool to scan and analyze the UI components of an Android application.To automate any android application using Appium, a user needs to identify the objects in AUT (Application under test). With "UIautomatorviewer" you can inspect the UI of an android application to find out the hierarchy and view different properties (id, text…) of the element.

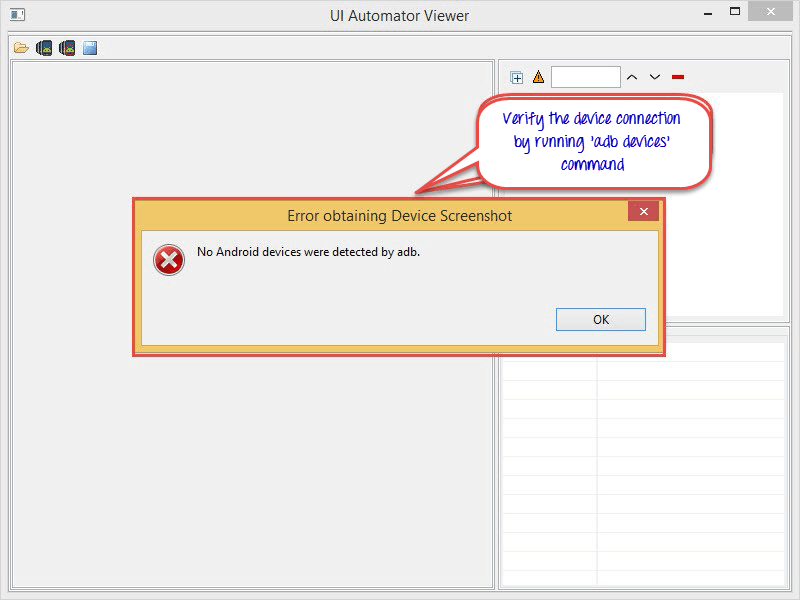
While executing automation scripts, Appium uses "Uiautomatorviewer" to identify different properties of the object and use the properties to identify the required object.

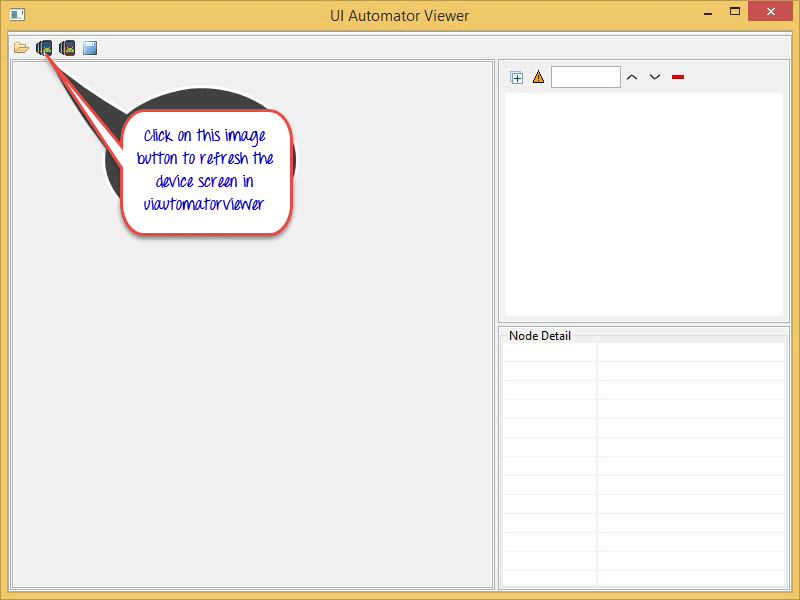
[](https://www.guru99.com/images/9-2015/082715_1132_Identifying1.png)

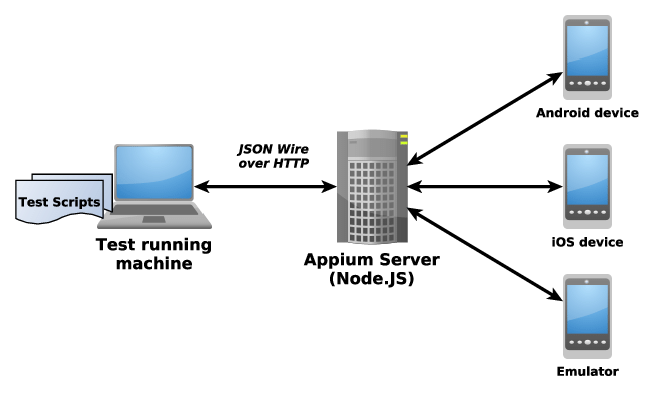
**Locator techniques:** id, classname, xpath, androiduiautomator, accessibilityid

**Path of uiautomatorviewer:** C:\Users\rahul.patidar\AppData\Local\Android\Sdk\tools\bin



**If device is not connected**

**Taking Screenshot of the current screen**

**Appium Architecture:** 

Appium Java Client (Eclipse-Where will write our code, internally these command will convert into JSON format(commands will go in the form of JSON payload) and client send session request to Appium Server and Server will respond Session Id to client for the connection and then connection will establish)

Server will connect with IOS (UI Automation framework developed by Apple)/Android (UI Automator developed by Google) and then this connection will be happen to Real device/Emulator/Simulator.

### **Types of Mobile apps:** Types of Mobile AppsTypes of Mobile Applications We Can Develop in 7th Media Digital ...**Native Apps:**

Native mobile apps are exclusively built for a specific type of Operating system. They are called  
native because they are native to a particular device or platform. Apps built on one [type of operating system](https://www.pcloudy.com/android-and-ios-basics-and-comparison/) cannot be used on another OS. In other words, Android apps can’t be used on the iPhone. They use the development tools and language that the respective platform supports (e.g., [Xcode](https://www.pcloudy.com/xcuitest-for-ios-apps-and-how-to-test-with-xcode/) and Objective-C with iOS, [Eclipse](https://www.pcloudy.com/pcloudy-plugin-for-eclipse-ide/), and Java with Android. It provides full access to all device controls like contacts, camera, sensors, etc. The native apps ensure high performance and great user experience as the developers use the native device UI. Native apps can be accessed via respective app stores eg- Android apps on Google Play Stores, iOS apps on App Store, etc.

#### **Advantages of native apps:**

1. Natives are very fast.  
2. Easily distributed in google apple app stores.  
3. More interactive and intuitive.  
4. Easily interact with any feature of the phone.

#### **Disadvantages of native apps:**

1. Built for a single platform  
2. Languages like swift and java used to build these types of apps are hard to learn.  
3. Expensive to develop.  
4. hard to maintain.

#### **Examples of Native Apps:**

  
   
  
Native apps are built using the native device operating system APIs and SDKs. These are coded using a platform specific language like Objective C for iOS, Java for Android, and C# for Windows phone. One can use the standard GUI components that are part of the platform SDK, easily creating a look and feel that is native to the OS and straightforward.  
   
These apps can access all the device hardware including the various sensors and peripherals if any. These apps are quite fast since the executable is compiled for the specific OS and are run directly on the OS. These come with their development environments including various simulators and infrastructure to do actual device testing.

### **Mobile Web Apps**

These are the web applications to deliver web pages on [web browsers](https://www.pcloudy.com/5-reasons-why-testing-is-incomplete-without-cross-browser-tests/) running on mobile devices. These are web-based mobile apps that do not get installed on your handheld mobile device and are run on web-hosted servers. Mobile web apps typically use HTML, CSS, Javascript, JQuery web technologies. They cannot access all features of native device functionality(camera, calendar, geolocation, etc.).

#### **Advantages of web apps:**

1. Reduced business cost.  
2. No installation needed.  
3. Better reach as it can be accessed from anywhere.  
4. Always up-to-date.

#### **Disadvantages of web apps:**

1. Dependent on internet speed.  
2. Interface not that sophisticated.  
3. Take a longer time to develop.  
4. Security risk.

#### **Examples of Mobile Web Apps:**



### **Hybrid Apps:**

Hybrid apps are the mixtures of native and mobile web apps. Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app. These are developed using technologies like HTML, CSS, Javascript, JQuery, Mobile Javascript frameworks, Cordova/PhoneGap etc. Like Native apps, Hybrid apps are also installed in the device and distributed through the app store. These are good for building apps that do not have a [high-performance](https://www.pcloudy.com/mobile-app-performance-monitoring-basics-to-advanced/) requirement but need full device access.

#### **Advantages of hybrid apps:**

1. Easy to build  
2. Much cheaper than a native app  
3. Single app for all platforms.  
4. No browser needed  
5. Can usually access device utilities using an API  
6. Faster to develop than native apps.

#### **Disadvantages of Hybrid apps:**

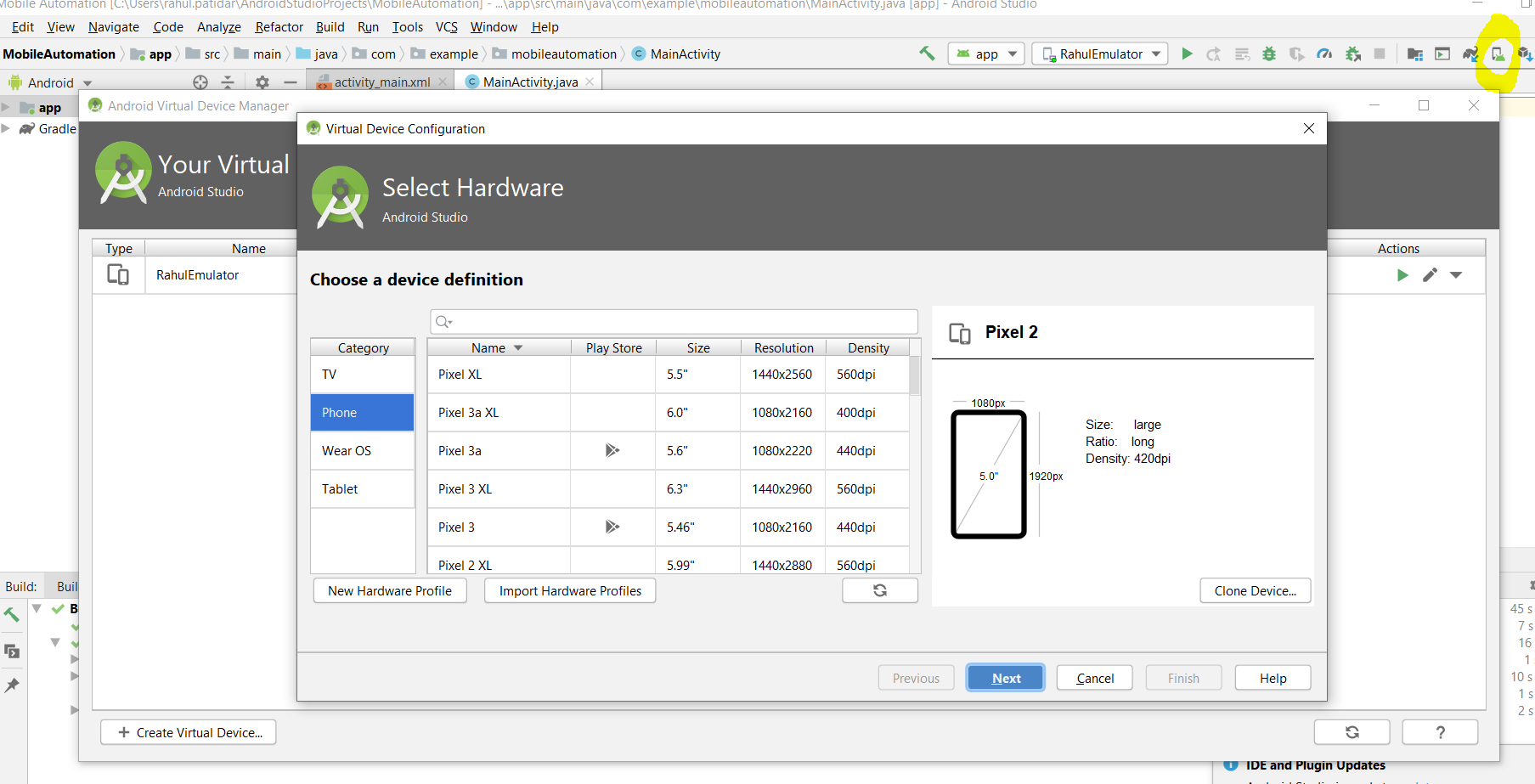
1. Slower than native apps  
2. more expensive than web apps  
3. Less interactive than native apps

#### **Examples of Hybrid Apps:**



**Android Device Manager (AVD Manager):**

You use the Android Device **Manager** to create and configure Android Virtual Devices (AVDs) that run in the Android Emulator.



Once Emulator is created you can directly open from Android studio **OR** go to emulator location in android sdk (C:\Users\rahul.patidar\AppData\Local\Android\Sdk\tools) and type

"**emulator -avd virtualDeviceName**" and enter, device will open.

## Appium Desired Capabilities :

Desired Capabilities are keys and values encoded in a JSON object, sent by Appium clients to the server when a new automation session is requested (Session takes 60sec to close the session but connection will still open and if we wants to close before 60sec than we need to stop the connection and start the connection). They tell the Appium drivers all kinds of important things about how you want your test to work. Each Appium client builds capabilities in a way specific to the client's language, but at the end of the day, they are sent over to Appium as JSON objects.

Desired Capabilities can be scripted in the WebDriver test or set within the Appium Server GUI (via an Inspector Session)

**Toast message:** Popup will come for a second or 2 sec

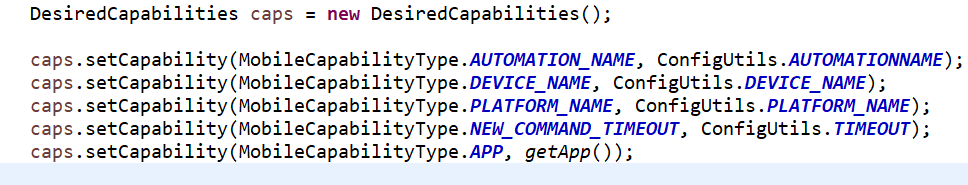
Android developer always use: android.widget.Toast as a class for those messages and most of the time you will not able to locate, so we have to use this class name for xpath.

android.widget.Toast

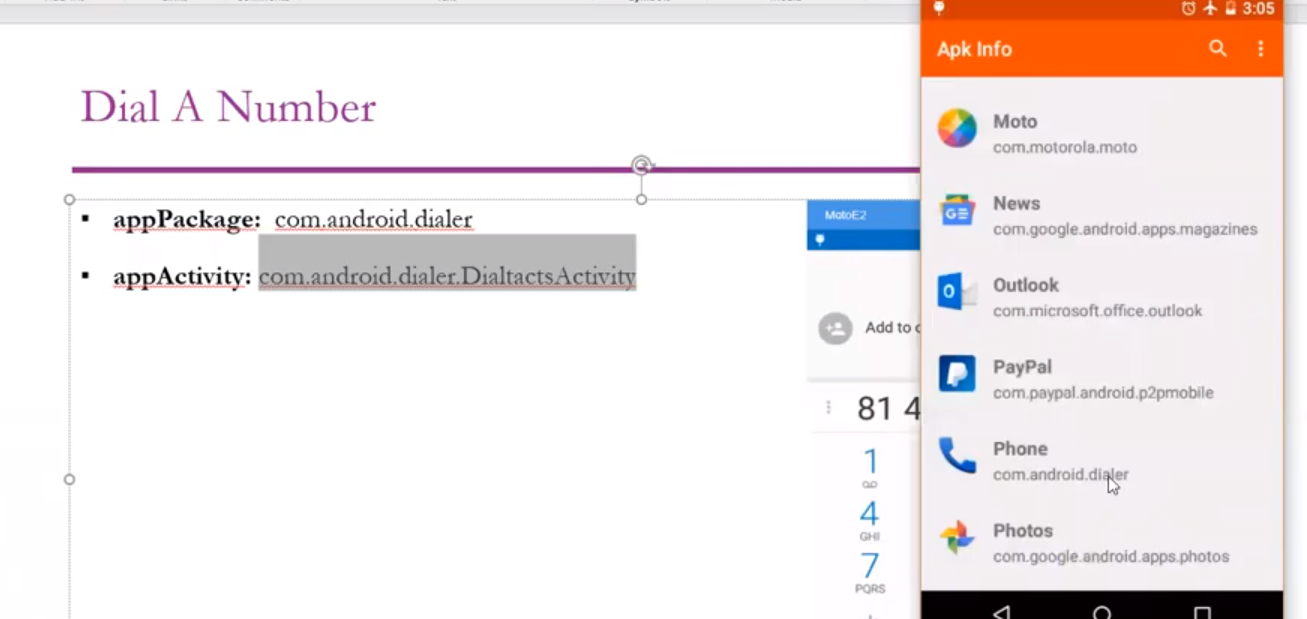
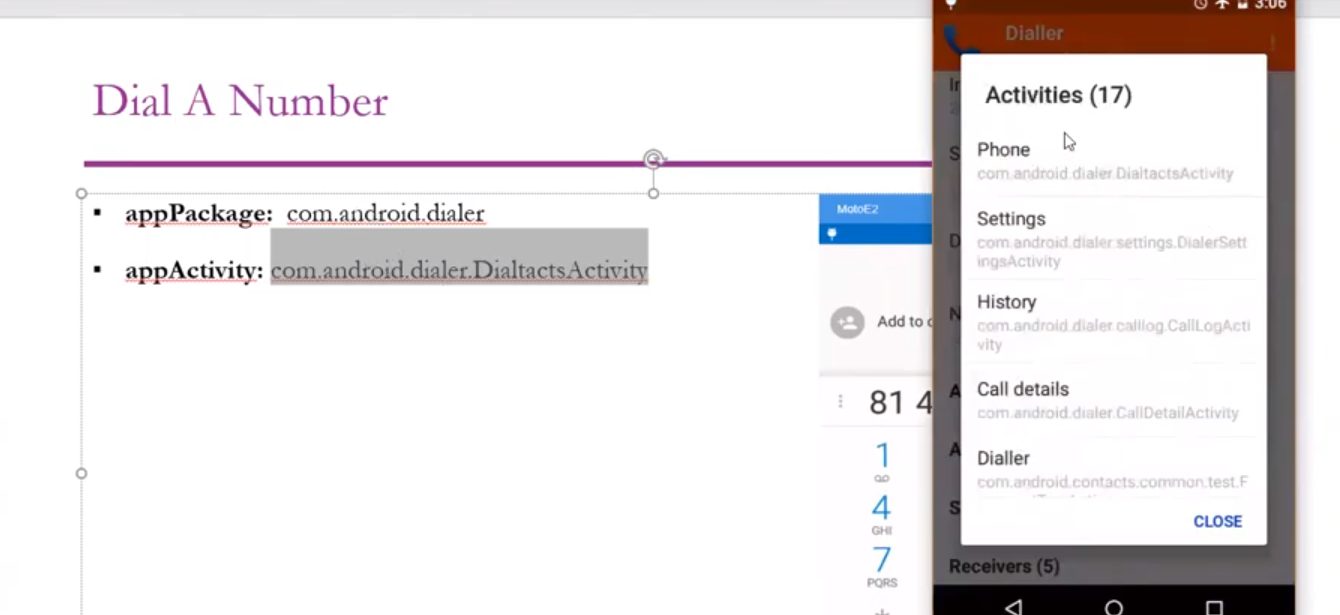
android.widget.Toast[1]

android.widget.Toast[2]

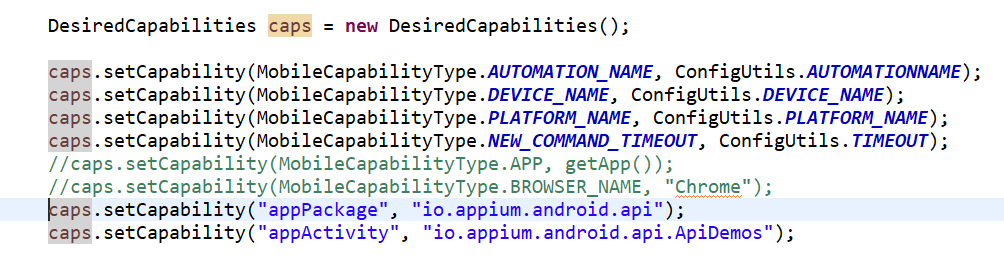
**DesiredCapabilities for Real(Android Device)/Virtual Device(RahulEmulator):**

 Android updated its internal framework to uiautomator2 and through Appium code, we need to tell that we need to access uiautomator2 elements of Android.

If we don’t have any apk file and app is already installed in mobile(example: calculator) now will check appPackage and appActivity using ApkInfo app(downloaded from play store) and will define appPackage and appActivity in desired Capabilities in place of App.

**Example:**



**DesiredCapabilities for Browser:**

