



Mobile Automation Testing and Appium

21TH JUN, 2023

(expleo)

Contents

- Introduction
- What is Appium
- Prerequisites for using Appium
- Benefits of Appium
- Appium Architecture
- Mobile App Testing using Emulator
- Mobile App Testing using Real Device

Introduction

- Mobile automation testing is the use of automation tools to test new mobile applications before they are commissioned for use.
- This type of testing is done to check whether mobile applications work well across various operating systems and mobile devices.
- Automation in testing increases the speed and accuracy of testing. It is an effective way of checking defects in new mobile applications.
- This testing performs various aspects of testing such as usability, security, performance, quality, and functionality.
- It also tests the ability of a new application to interact with other applications.

What is Appium

What is Appium?

- Appium is an automation testing tool that can be used to validate mobile browsers and mobile applications.
- This tool is widely used in mobile automation testing because it is free and can support both iOS and Android platforms.
- Mobile apps have become an important element in our lives because they provide us a smart way of doing things like shopping, paying bills, chatting, and booking flights.

What is Appium

What is Appium?

- Appium is a popular mobile automation testing tool used for testing mobile browsers and mobile applications. It validates the compatibility, usability, and response time of mobile browsers and applications across various mobile devices.
- This software has been limited to mobile application testing, with a specific focus on iOS, and Android applications. Various programming languages are supported in Appium automation testing. These include Python, PHP, Java, and Perl.
- Automation tests in Appium can be run on mobile devices, simulators, and emulators.
- A simulator is an imitation of the condition and operation of an application. Similarly, an emulator is software or hardware that allows a computer system (host) to run applications designed for another system (guest).

What is Appium

What is Appium?

- Automation testing in Appium does not depend on the operating system (OS) of a mobile device. This is because its framework can translate the driver commands into Android or iOS commands that do not depend on the type of the OS. Instead, these commands depend on the type of mobile device.
- The following are some of the main types of mobile apps that can be tested using Appium automation testing:
- Native apps: These are platform-specific apps that are developed and operated in Android, iOS, or Windows SDKs. These apps do not have features for web browser navigation.

What is Appium

What is Appium?

- Hybrid apps: These are apps that can work well on a certain platform and still have features for web browser navigation. These are mobile applications containing features for web browser navigation.
- Web apps: These are apps that can be accessed using browsers (built-in) of mobile devices.

What is Appium

Prerequisites for using Appium

- To use Appium, you need the following prerequisites:
 1. Node.js
 2. Java
 3. Android Studio
 4. Eclipse IDE
 5. Appium Server
 6. Appium Desktop Client
 7. Selenium Jar

What is Appium

Benefits of Appium

- Appium is widely used because of its unique features and the certain benefits it offers mobile app developers. The main features that make developers choose Appium over other mobile automation testing tools are:
- Open source: This tool is open source, which reduces the overall cost of developing and rolling out a mobile application for deployment.
- Cross-platform: Automation testing in Appium can be done on various platforms such as Windows, iOS, and Android. Other tools like Robotium and Selendroid support testing on only Android.
- Multiple mobile apps: Appium allows testing in various types of mobile applications (native, web, and hybrid apps).

What is Appium

Benefits of Appium

- Multiple programming languages: Various programming languages are supported in Appium automation testing. These include Ruby, Javascript, C#, Python, and Java. This gives testers and developers some flexibility on the choice of language.
- Integration: This tool can be used with other external applications such as Selenium Grid and Selenium WebDriver.
- Low memory consumption: The architecture of Appium functions as a proxy between the tool-kit for automation and the test machine.

What is Appium

Benefits of Appium

Appium is also chosen by testers and developers because it offers the following additional benefits:

- It allows the test scripts to be executed simultaneously.
- It contains a recording tool to record test cases and a playback tool for re-running such records.
- Various elements can be tested simultaneously (i.e. emulators, real devices, and simulators).
- It supports cloud-based testing using testdriod.
- Appium is backed by an active community (Google group) that improves the ease and speed of troubleshooting.

What is Appium

Appium Architecture

The Appium architecture consists of three main components:

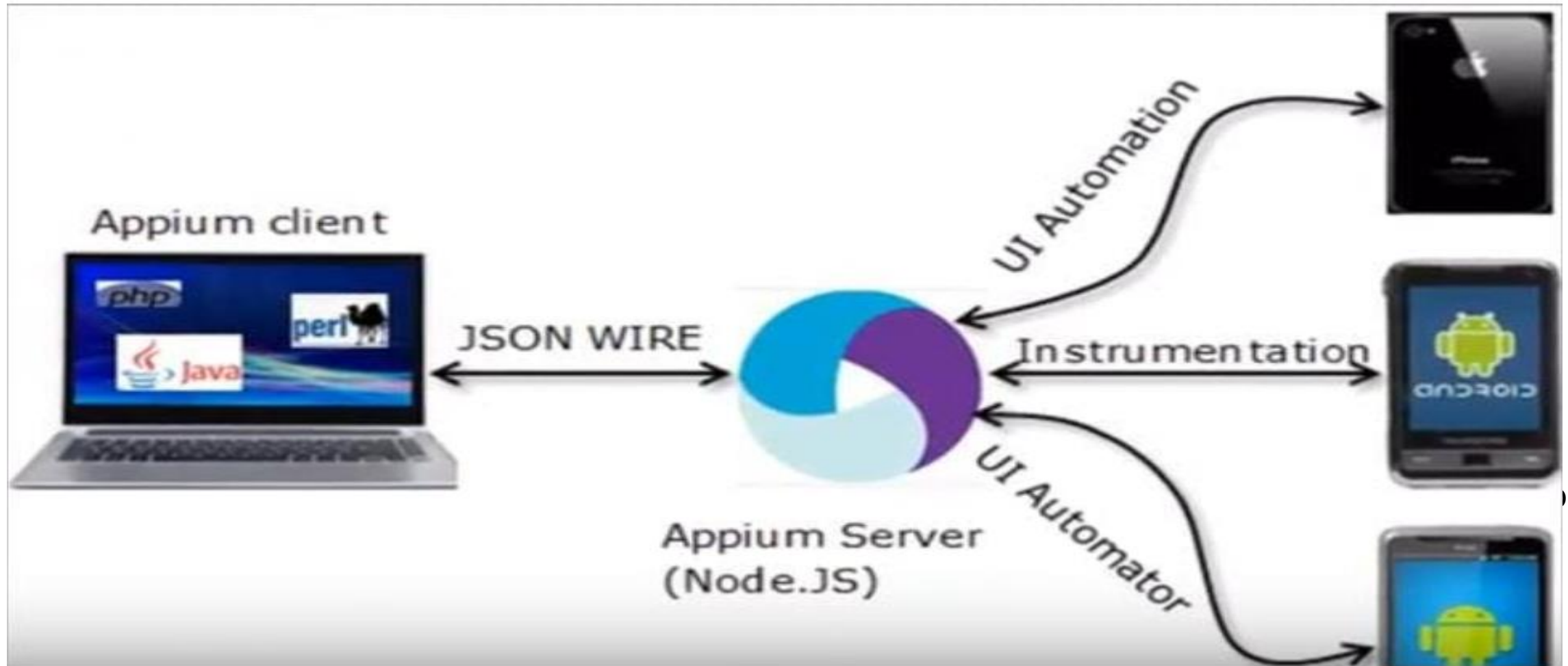
- 1.Appium Client
- 2.Appium Server
- 3.End device

Appium client

This contains the automation code (scripted). This code is written in a specific programming language like Java, Python, Ruby, or Perl. The script contains the configuration details of the application and the mobile device. The scripted code and the configuration details are used to run the applications' test cases.

What is Appium

Appium Architecture



What is Appium

Appium Architecture

Appium server

This is an HTTP server that receives command requests (HTTP requests) from the client component (in JSON format). This server uses a Node.js server. The commands are then executed on mobile devices.

This component should be installed on the computer before scripting the automation code. The Appium server acts as an intermediary between the Appium Client and the end device. It is connected to platforms such as iOS, Android, and Windows.

What is Appium

Appium Architecture

End device

This refers to an emulator or a mobile device that is connected to the Appium server. The test cases are executed on this device.

- Appium is backed by an active community (Google group) that improves the ease and speed of troubleshooting.

What is Appium

How does Appium Works?

- Appium works using the three main components of its architecture (client, server, and end device).
- When this tool is installed on your computer, it sets up an HTTP server that generates a REST API to enhance communication with the client.
- This server collects command requests from the Appium client in JSON format.
- JSON Wire Protocol is a medium of communication between the server and the client.
- It plays the role of communicating command requests. These command requests are then executed on the end devices (either on Android, Windows, or iOS).

What is Appium

How does Appium Works?

- The Appium server uses a test automation framework to execute requests on the user interface of end devices. For example, it uses the XCUI test to execute commands on iOS. It uses UI Automator to execute command requests on Android platforms (for newer API versions).
- The XCUI test allows users to write tests on iOS using Objective-C or Swift. The UI Automator is an Android testing framework that builds user interface (UI) tests using APIs. After the requests have been executed, the Appium server provides HTTP responses to the client.
- Appium works differently on iOS and Android. Let's look at how it works on these two platforms.

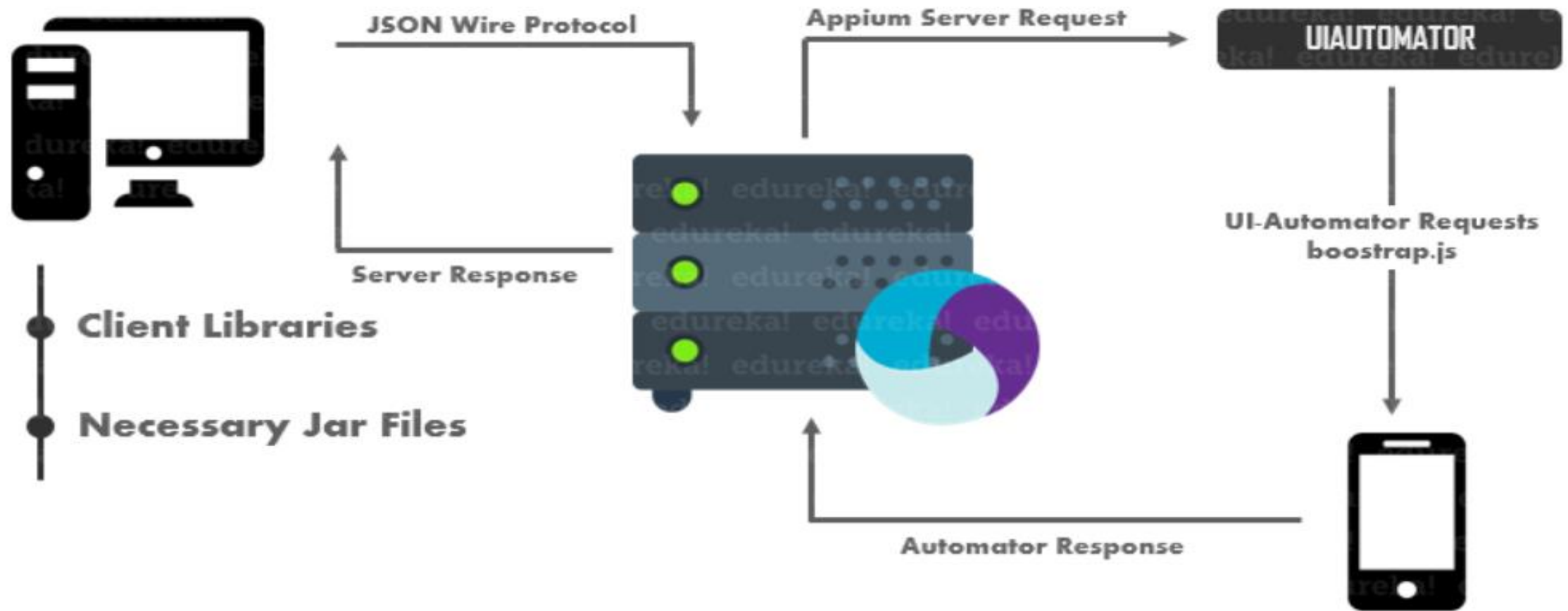
What is Appium

How does Appium Work on Android?

- 1.The Appium client connects with the HTTP server and uses the JSON Wire Protocol for communication.
- 2.The Appium server receives requests from the client component.
- 3.The server connects with the mobile automation framework for Android (UI Automator or Selendroid).
- 4.Android devices contains bootstrap.jar that receives the test commands from the Appium server. The bootstrap.jar contains executable files that enhance the connection between the server and Android devices. It plays the role of a TCP server since it enhances the secure transmission of test commands to the Android devices.

What is Appium

How does Appium Work on Android?



What is Appium

How does Appium Work on Android?

5. The bootstrap.jar uses the UI Automator or Selendroid to execute the command requests on the Android device.
6. The test results are then sent to the server, which eventually sends HTTP responses to the Appium client. The HTTP responses contain status codes indicating success or server error.

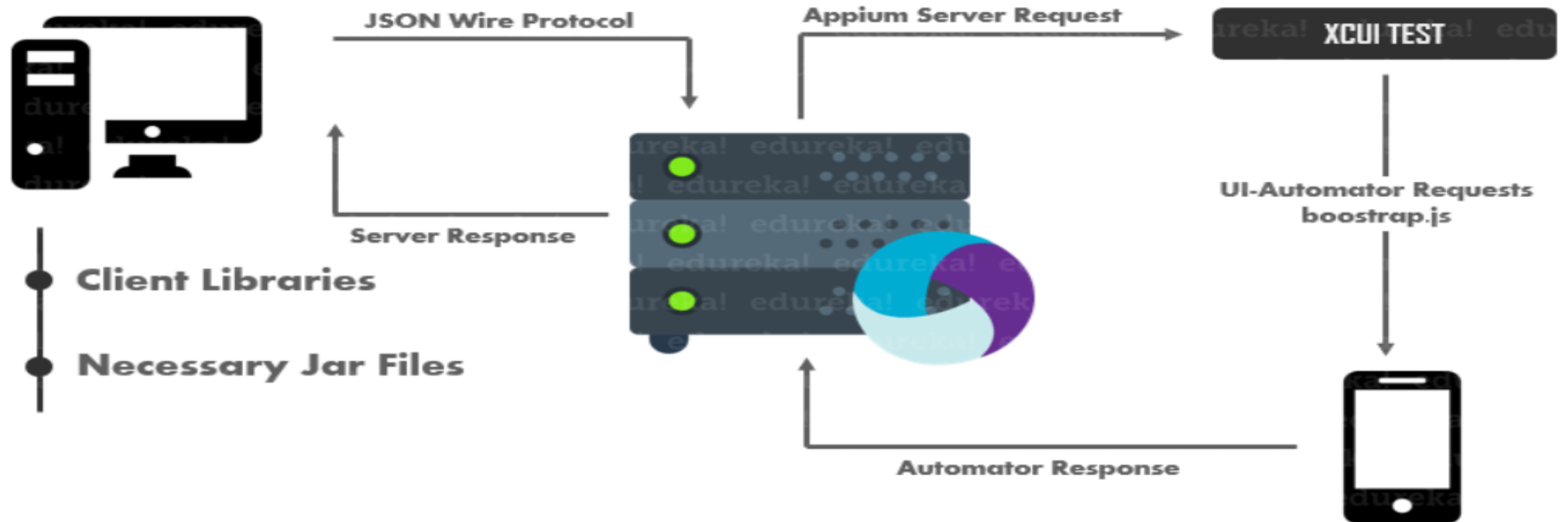
What is Appium

How does Appium Work on iOS?

- 1.The Appium client connects with the HTTP server and uses the JSON Wire Protocol for communication.
- 2.The Appium server receives requests from the client component.
- 3.The server connects with the mobile automation framework for iOS devices (XCUI test).
- 4.iOS devices contains bootstrap.js that receives the test commands from the Appium server.
The bootstrap.js contains executable files that enhance the connection between the server and the iOS devices. It plays the role of a TCP server since it enhances the secure transmission of test commands to the iOS devices.

What is Appium

How does Appium Work on iOS?



What is Appium

How does Appium Work on iOS?

5. The bootstrap.js uses the XCUI test to execute the command requests on iOS device.
6. The test results are then sent to the server, that eventually sends HTTP responses to the Appium client. The HTTP responses contain status codes indicating success or server error.

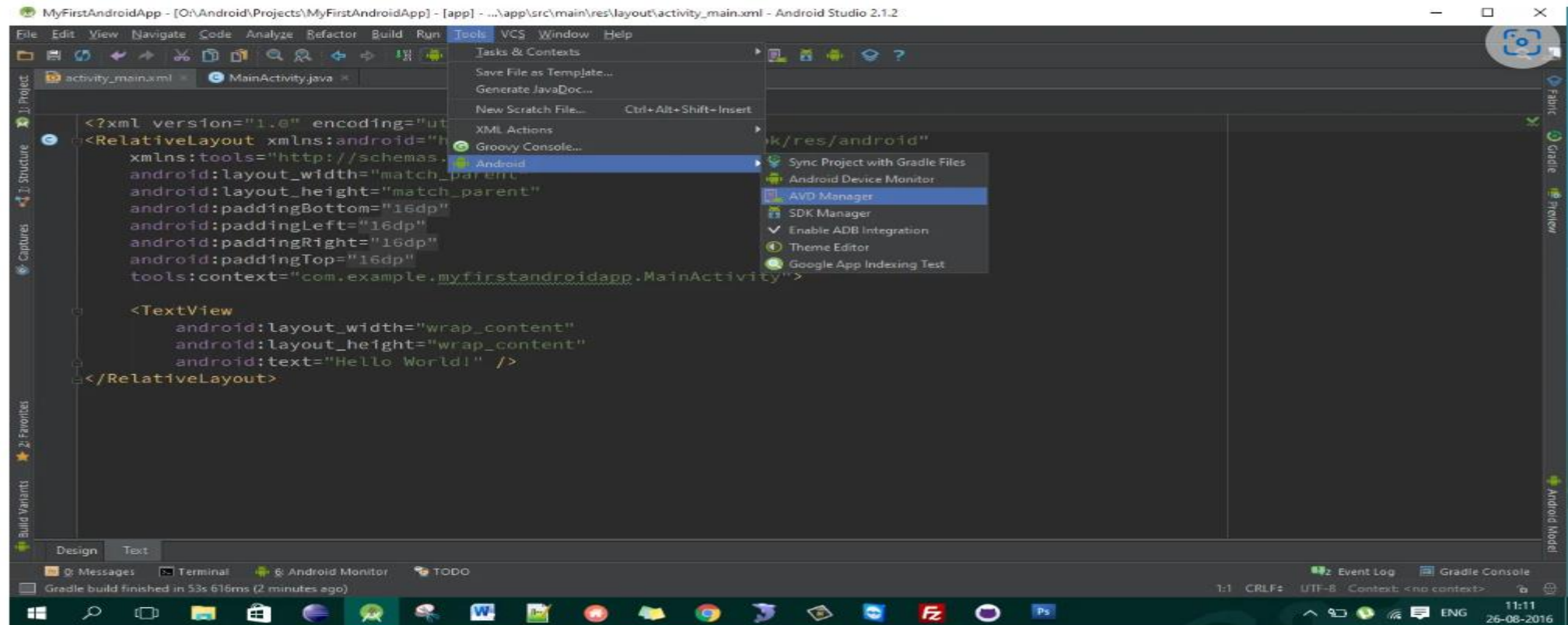
Step by Step process to setup Android Emulator on Windows

1. Download the Android Studio and Install.
2. Install SDK Tools from Android Studio from SDK Manager
3. Set Environment variables for SDK tools.
4. Create the Emulator from AVD Manager

Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager

Step 1: To open AVD manager, go to Tools → Android → AVD Manager as shown in below image.



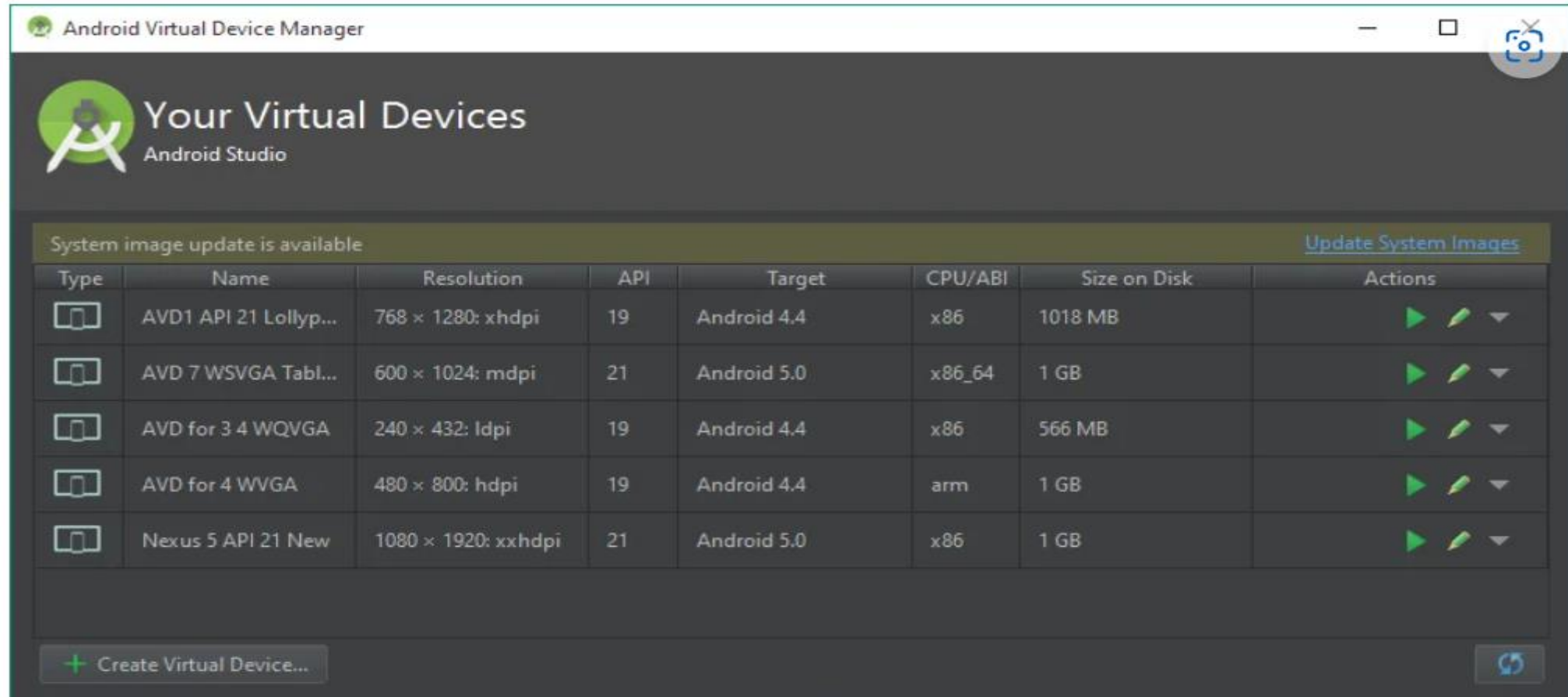
Creating an Android Virtual Device using Android Studio with AVD Manager

- It will open AVD Manager with a list of created virtual devices.
- It may be empty for you now as you haven't created any device as of now.

Step 2: To create a new device, click on Create Virtual Device button at the bottom-left corner.

Mobile App Testing using Emulator

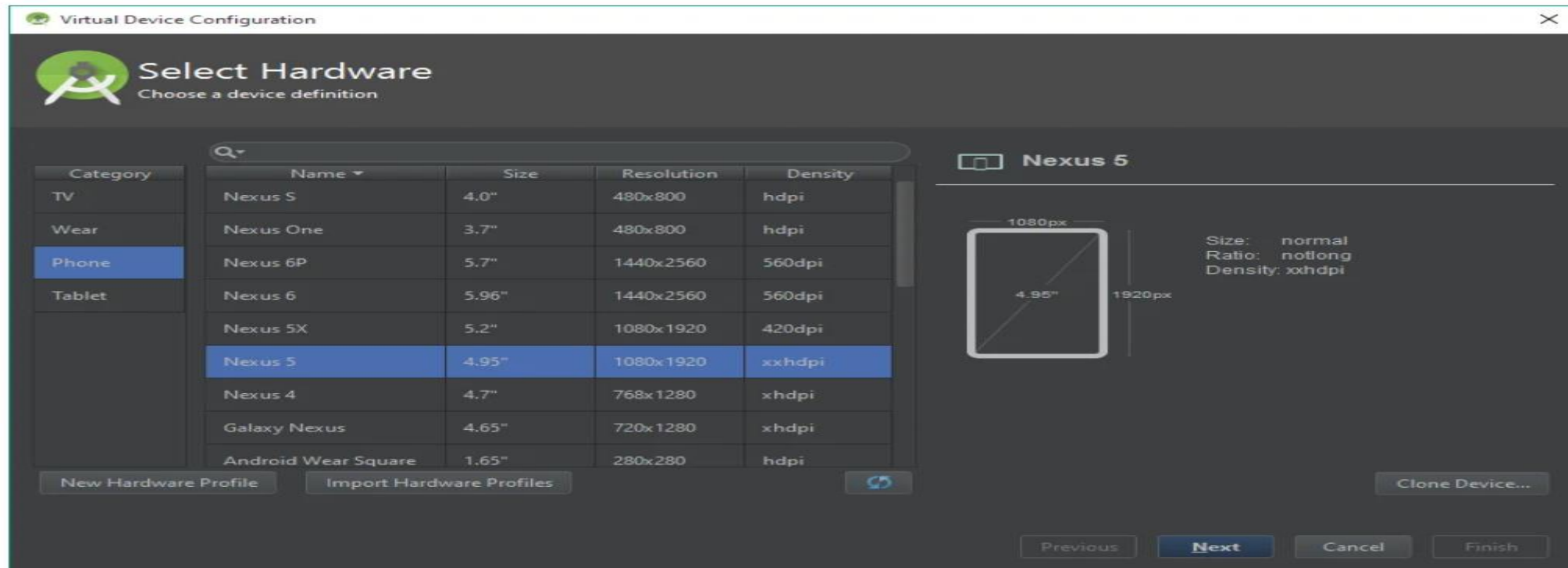
Creating an Android Virtual Device using Android Studio with AVD Manager



Creating an Android Virtual Device using Android Studio with AVD Manager

Step 3: Select any one out of all the devices listed, with your required configuration (like Size of the screen, Resolution and Density) and click on Next.

Note: It will open a window to Select Hardware type for your virtual device. This list contains almost all the Android devices with their respective settings.



Creating an Android Virtual Device using Android Studio with AVD Manager

Step 4: Next you will be asked to select System Image that will be the running Android Version for your newly created virtual device.

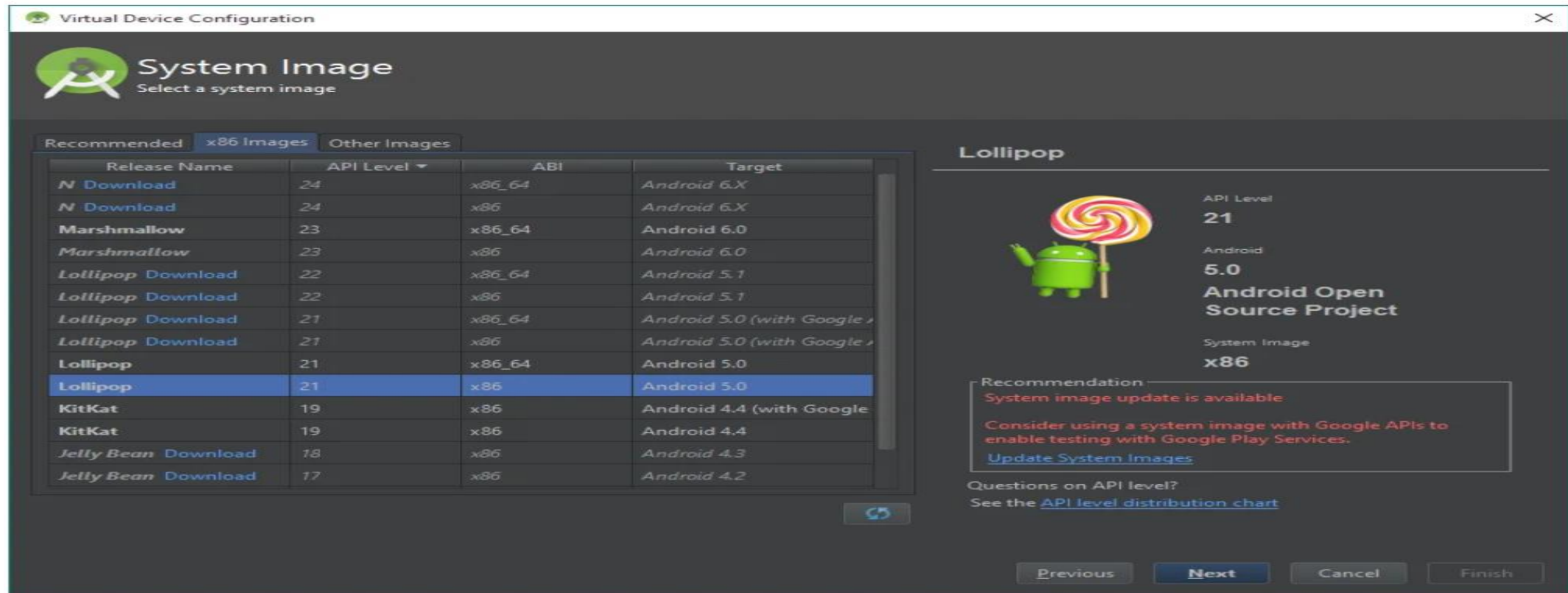
Note: You can choose any Android system images that are already available in your Android Studio, or Download the one you want, by clicking on the Download option available with the names.

- Recommended section will list the best choices available as per the latest updates available.
- x86 Images contain images that are mostly used and Other Images section contain system images with Google Play Services.
- Choose as per your required configuration (We've selected API level 21).

Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager

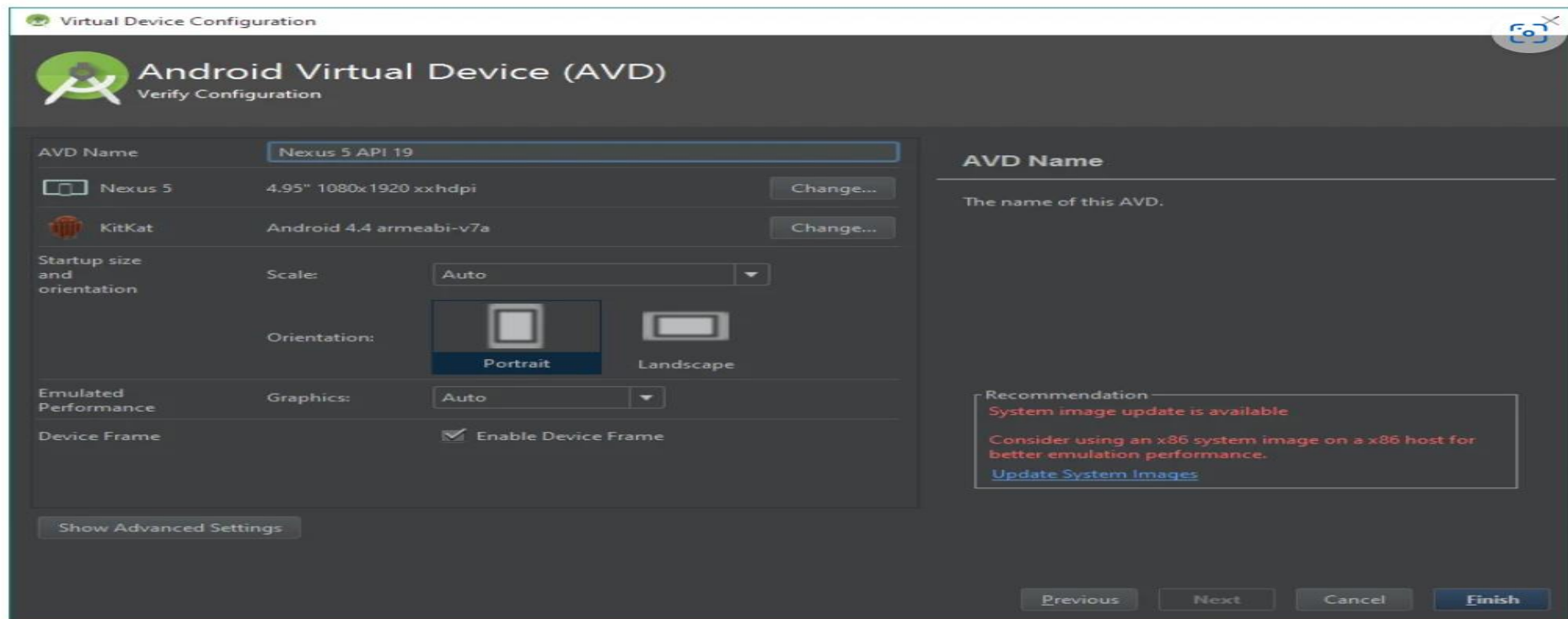
Step 4: Next you will be asked to select System Image that will be the running Android Version for your newly created virtual device.



Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager

Step 5: Next window will list down all the configured settings for final verification. Here, you can give your AVD a name for identification, can change device type and API configuration and can also setup size, orientation as well as Graphics for your AVD.



Creating an Android Virtual Device using Android Studio with AVD Manager

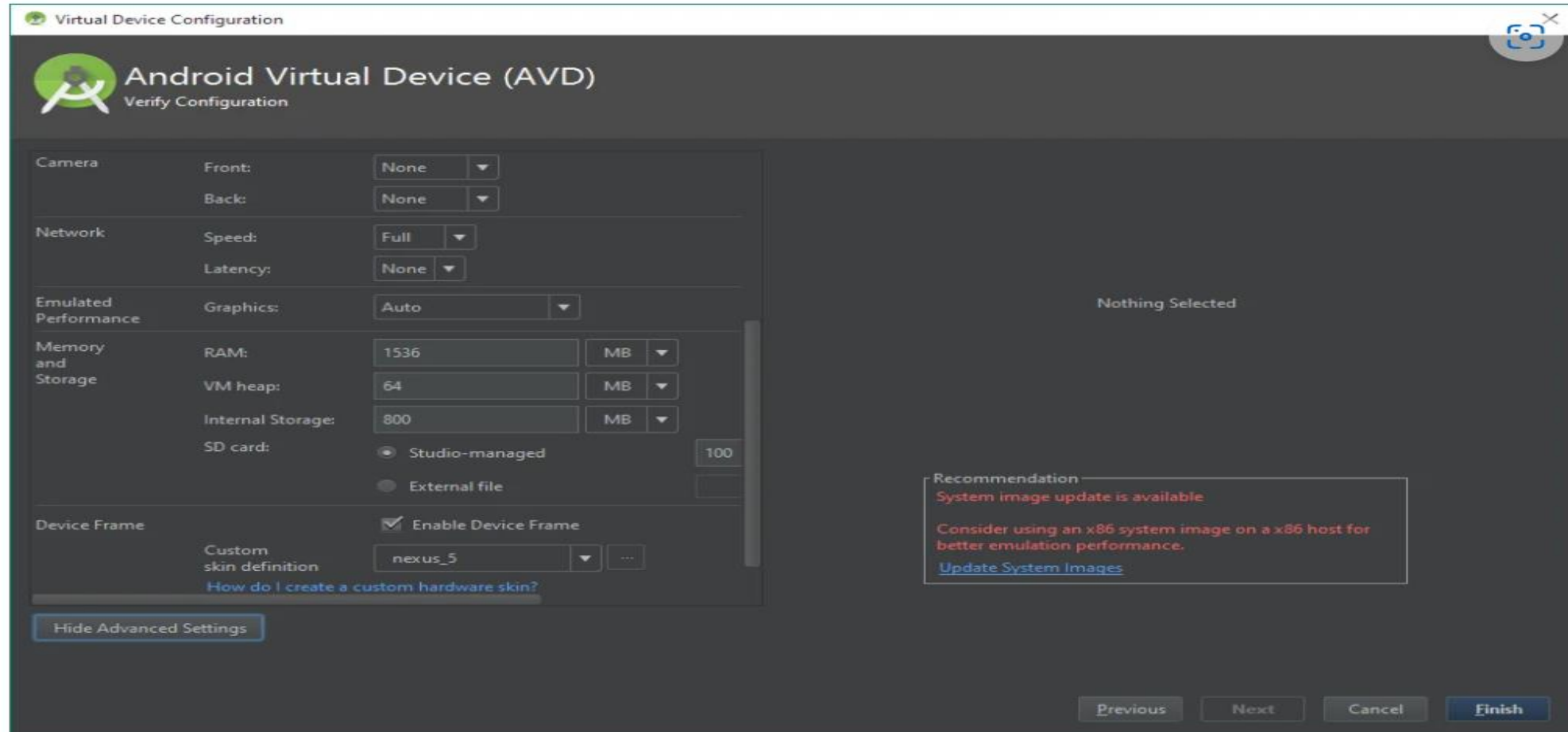
Step 6: Click on Show Advanced Settings and you will see more advanced settings for your virtual device as shown in image below.

Here you have settings for Camera, Network, Memory (RAM & Heap) and Storage (Internal & External) and Virtual Device Frame.orientation as well as Graphics for your AVD.

Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager

Step 6: Click on Show Advanced Settings

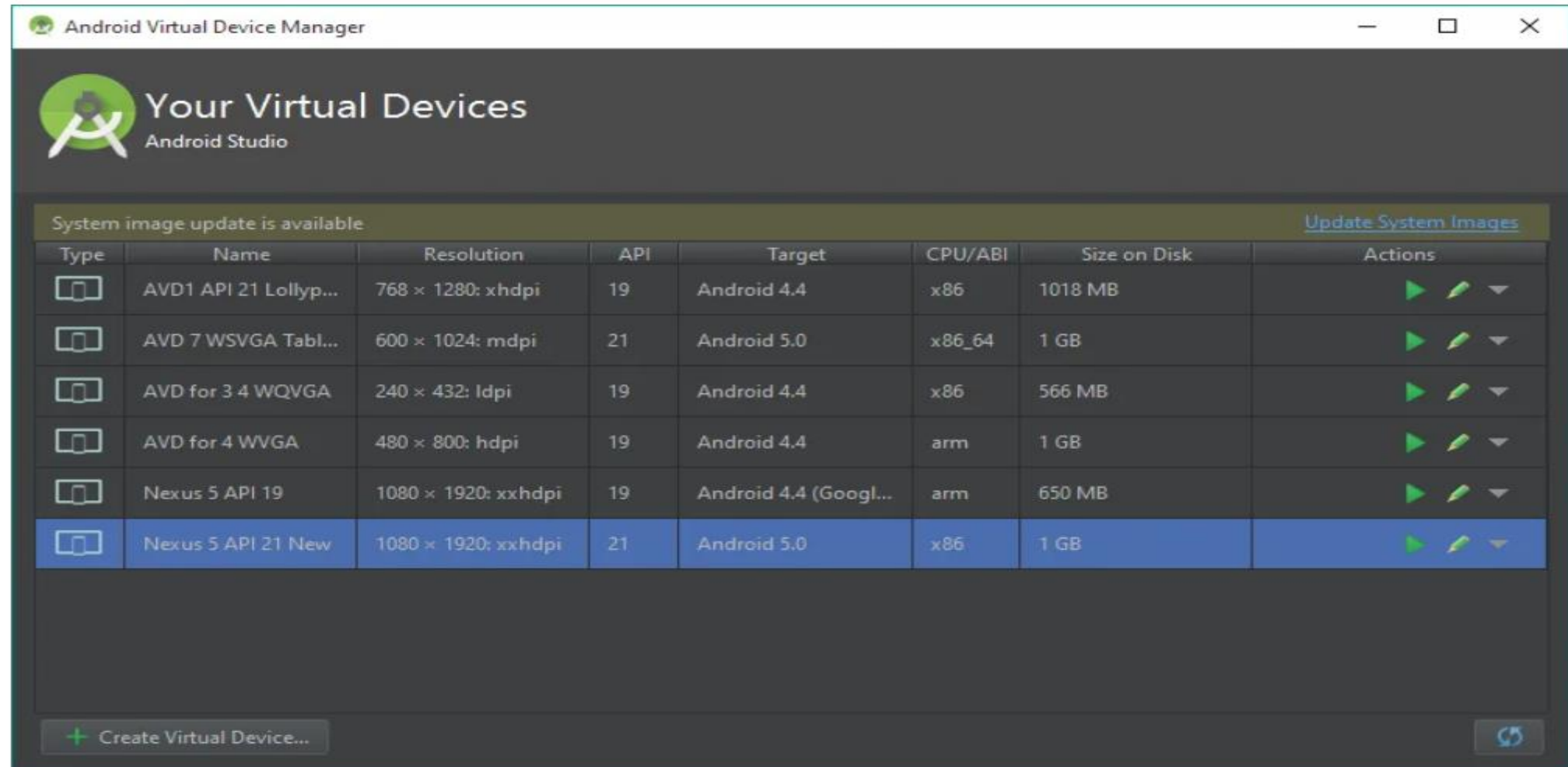


Creating an Android Virtual Device using Android Studio with AVD Manager

- You can configure your device as per your requirements and click on Finish.
- Android Studio will immediately start building AVD with the selected configurations & might take some time.
- When it completes, AVD Manager will list out your virtual device in the available devices list as shown in below image.

Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager

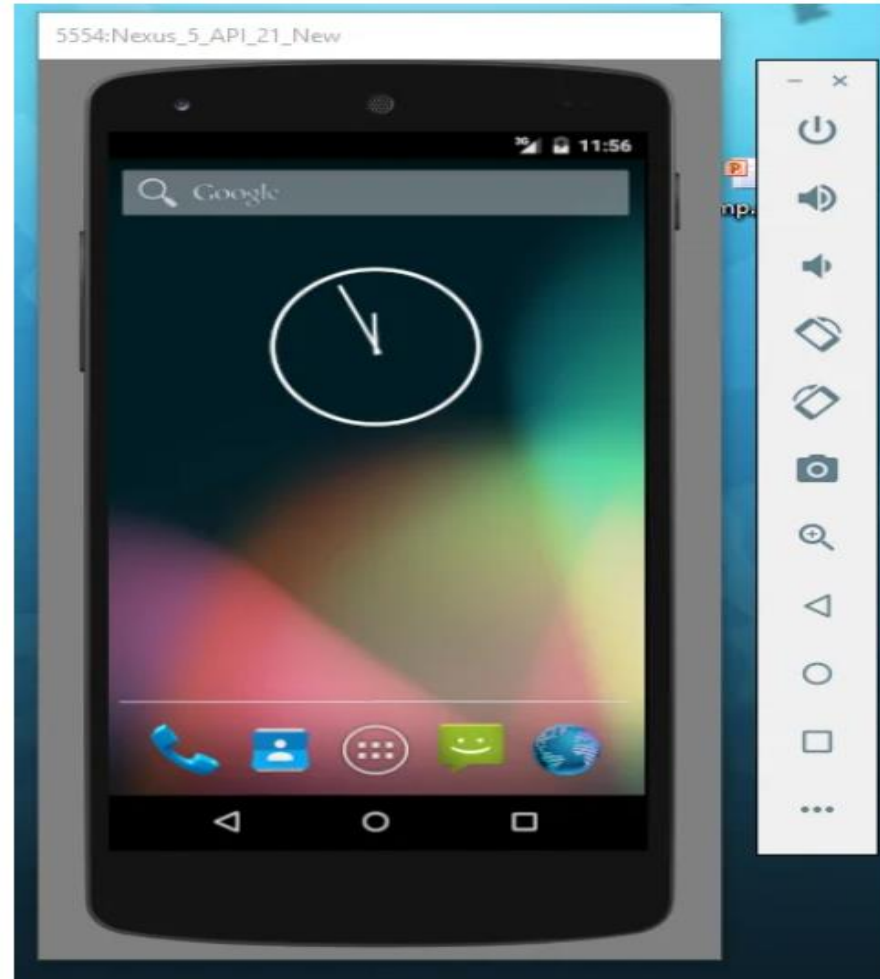


Creating an Android Virtual Device using Android Studio with AVD Manager

- From the Action column(last column of the table), you can perform several actions like Launch AVD and Edit AVD configurations etc.
- Launch your first AVD by clicking Start icon(green play icon).
- It will start a Virtual Device just like an Android Device as shown in below image.
- Side toolbar contains buttons to perform actions like volume up-down, change orientation, go back, go to home or recent & more.
- You can also turn the power off for the virtual devices using the power button and to close the virtual device select close button.

Mobile App Testing using Emulator

Creating an Android Virtual Device using Android Studio with AVD Manager



Mobile App Testing using Emulator

Appium Desktop Server






1. Download Appium Desktop Server from the link:

<https://github.com/appium/appium-desktop/releases/tag/v1.22.3-4>

2. Install by clicking the .exe file

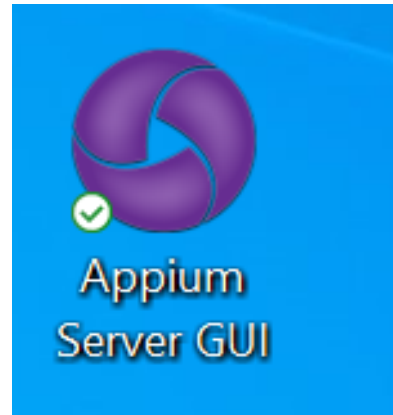
▼ Assets

12

 Appium-Server-GUI-1.22.3-4-mac.zip	161 MB	May 15, 2022
 Appium-Server-GUI-linux-1.22.3-4.AppImage	138 MB	May 15, 2022
 Appium-Server-GUI-mac-1.22.3-4.dmg	151 MB	May 15, 2022
 Appium-Server-GUI-mac-1.22.3-4.dmg.blockmap	162 KB	May 15, 2022
 Appium-Server-GUI-windows-1.22.3-4.exe	245 MB	May 15, 2022
 Appium-Server-GUI-windows-1.22.3-4.exe.blockmap	250 KB	May 15, 2022
 Appium-Server-GUI-windows-1.22.3-4.zip	154 MB	May 15, 2022
 Appium.Server.GUI-1.22.3-4-mac.zip.blockmap	171 KB	May 15, 2022
 latest-linux.yml	407 Bytes	May 15, 2022
 latest-mac.yml	533 Bytes	May 15, 2022
 Source code (zip)		May 15, 2022
 Source code (tar.gz)		May 15, 2022

Appium Desktop Server

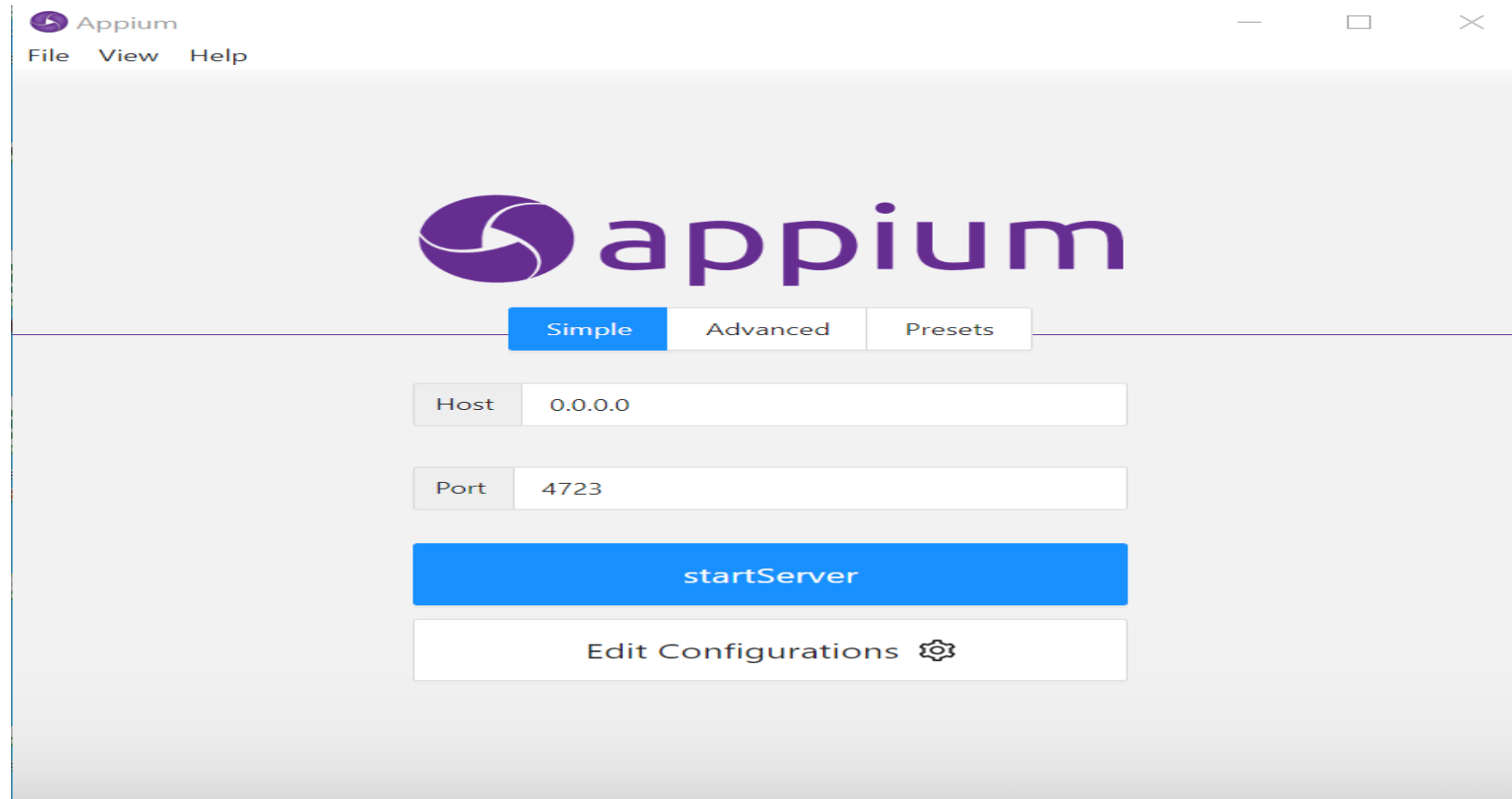
After successful installation you will see a desktop shortcut:



Mobile App Testing using Emulator

Appium Desktop Server

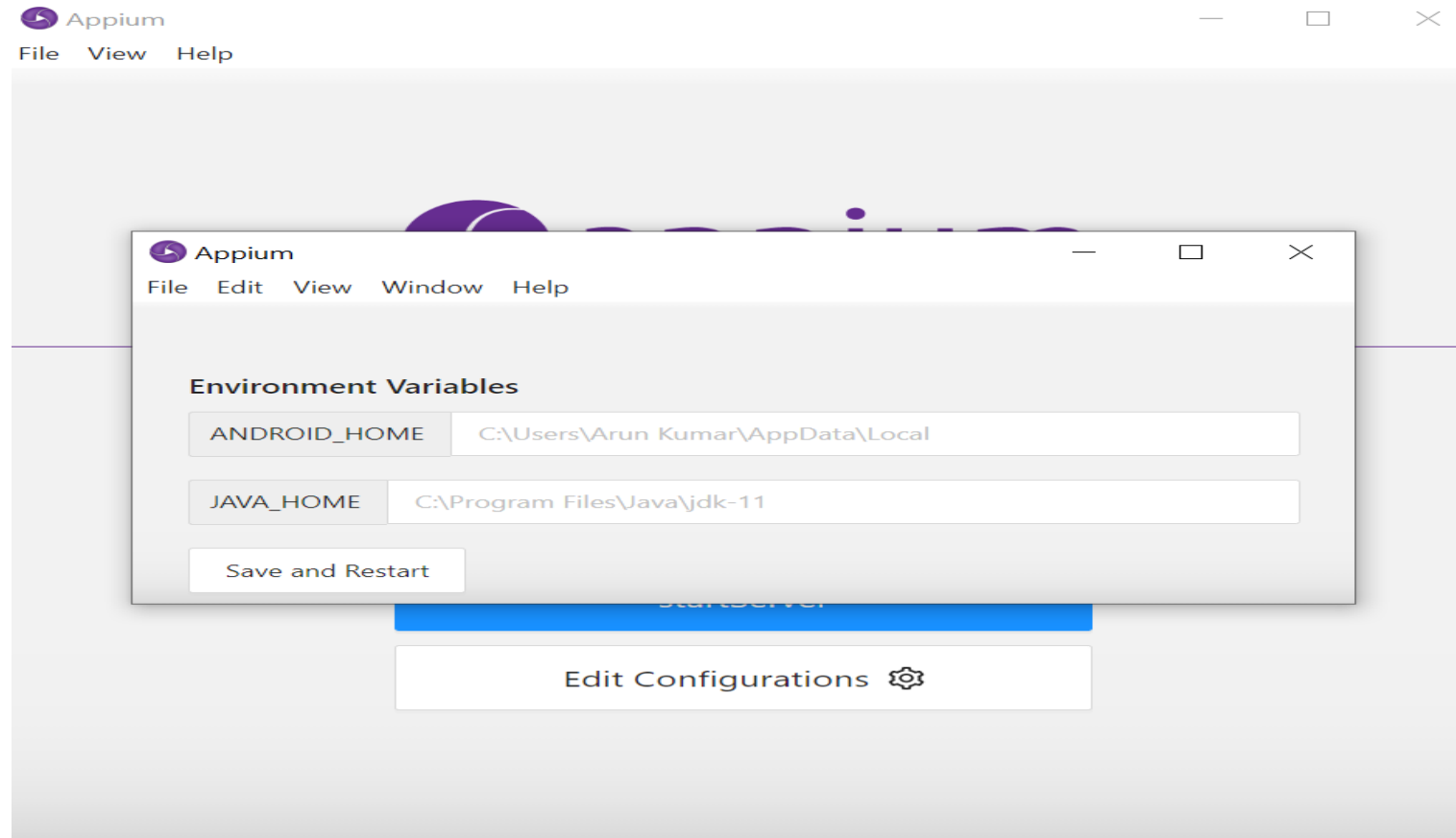
Double click to start the Appium Server whenever required



Mobile App Testing using Emulator

Appium Desktop Server

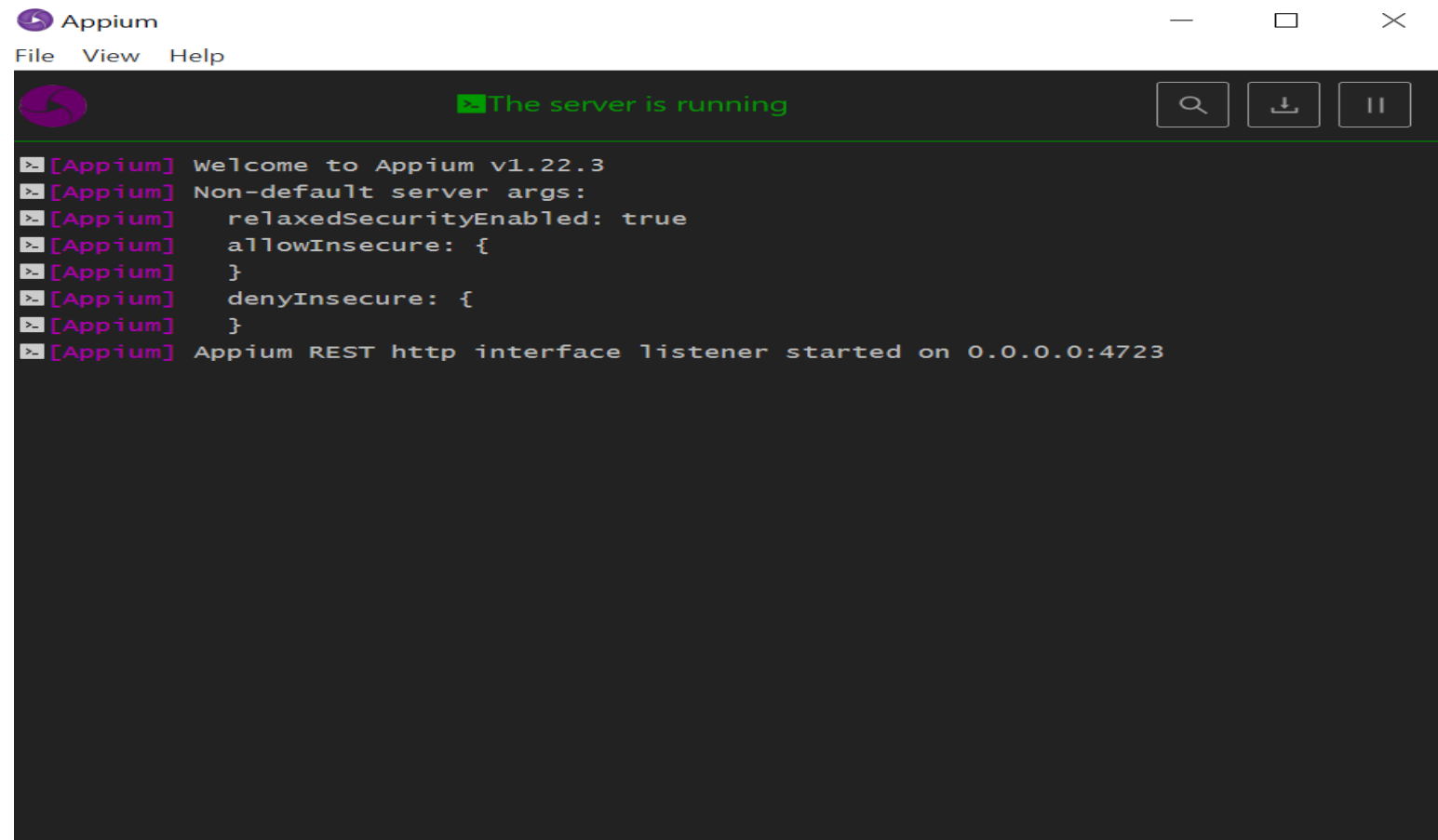
Check whether environment variables are set by clicking Edit Configuration.



Mobile App Testing using Emulator

Appium Desktop Server

Click start server to start the Appium Server. It will look like below once server started successfully.



The screenshot shows the Appium Desktop application window. The title bar says "Appium" with standard window controls. The menu bar includes "File", "View", and "Help". The main area has a dark background with a green status bar at the top that says "The server is running" with a green checkmark icon. Below the status bar, there is a log of messages from the Appium server, each preceded by a green prompt character and "[Appium]". The messages are: "Welcome to Appium v1.22.3", "Non-default server args:", "relaxedSecurityEnabled: true", "allowInsecure: {", "}", "denyInsecure: {", "}", and "Appium REST http interface listener started on 0.0.0.0:4723". On the right side of the main area, there are three buttons: a search icon, a download icon, and a pause icon.

```
> [Appium] Welcome to Appium v1.22.3
> [Appium] Non-default server args:
> [Appium]   relaxedSecurityEnabled: true
> [Appium]   allowInsecure: {
> [Appium]   }
> [Appium]   denyInsecure: {
> [Appium]   }
> [Appium] Appium REST http interface listener started on 0.0.0.0:4723
```

Mobile App Testing using Emulator

Appium Inspector





1. Download Appium Desktop Server from the link:

<https://github.com/appium/appium-inspector/releases/tag/v2023.8.4>

2. Install by clicking the .exe file

▼ Assets

12

 Appium-Inspector-2023.8.4-universal-mac.zip	181 MB	Aug 24
 Appium-Inspector-linux-2023.8.4.ApplImage	120 MB	Aug 24
 Appium-Inspector-mac-2023.8.4.dmg	189 MB	Aug 24
 Appium-Inspector-mac-2023.8.4.dmg.blockmap	204 KB	Aug 24
 Appium-Inspector-windows-2023.8.4.exe	168 MB	Aug 24
 Appium-Inspector-windows-2023.8.4.exe.blockmap	176 KB	Aug 24
 Appium-Inspector-windows-2023.8.4.zip	117 MB	Aug 24
 Appium-Inspector-2023.8.4-universal-mac.zip.blockmap	195 KB	Aug 24
 latest-linux.yml	405 Bytes	Aug 24
 latest-mac.yml	550 Bytes	Aug 24
 Source code (zip)		Aug 24
 Source code (tar.gz)		Aug 24

Mobile App Testing using Emulator

Appium Inspector

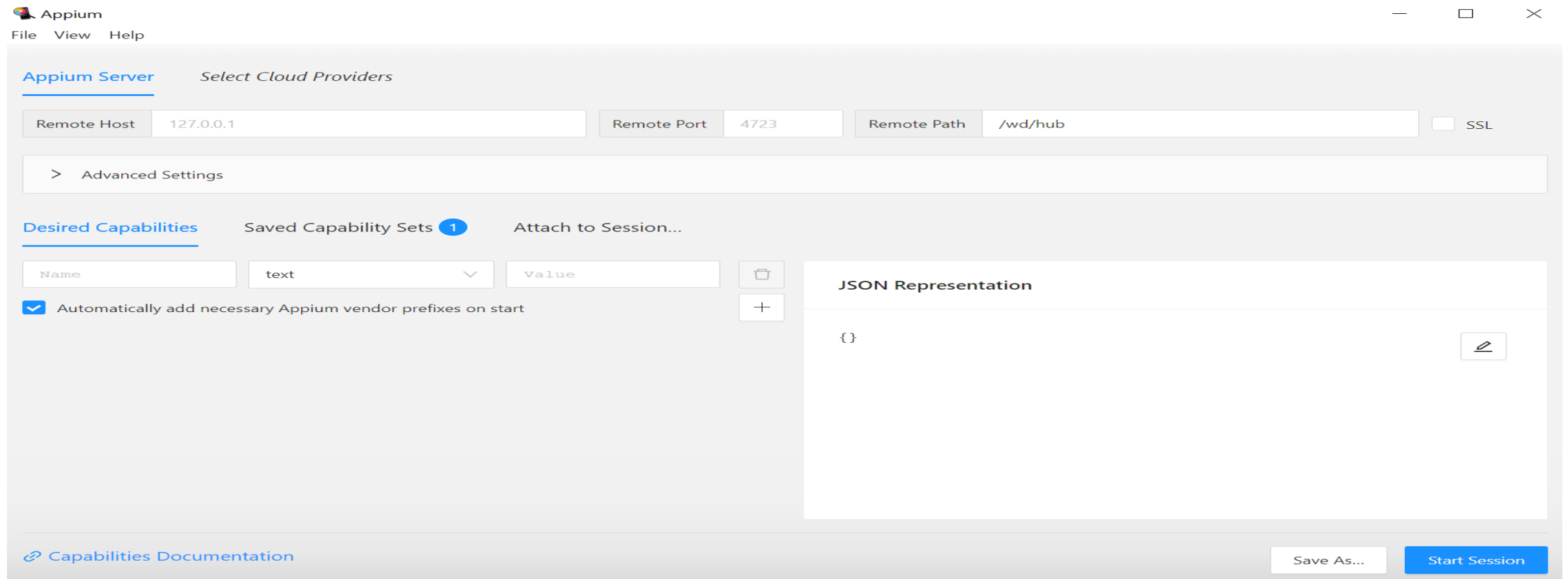
After successful installation you will see a desktop shortcut:



Mobile App Testing using Emulator

Appium Inspector

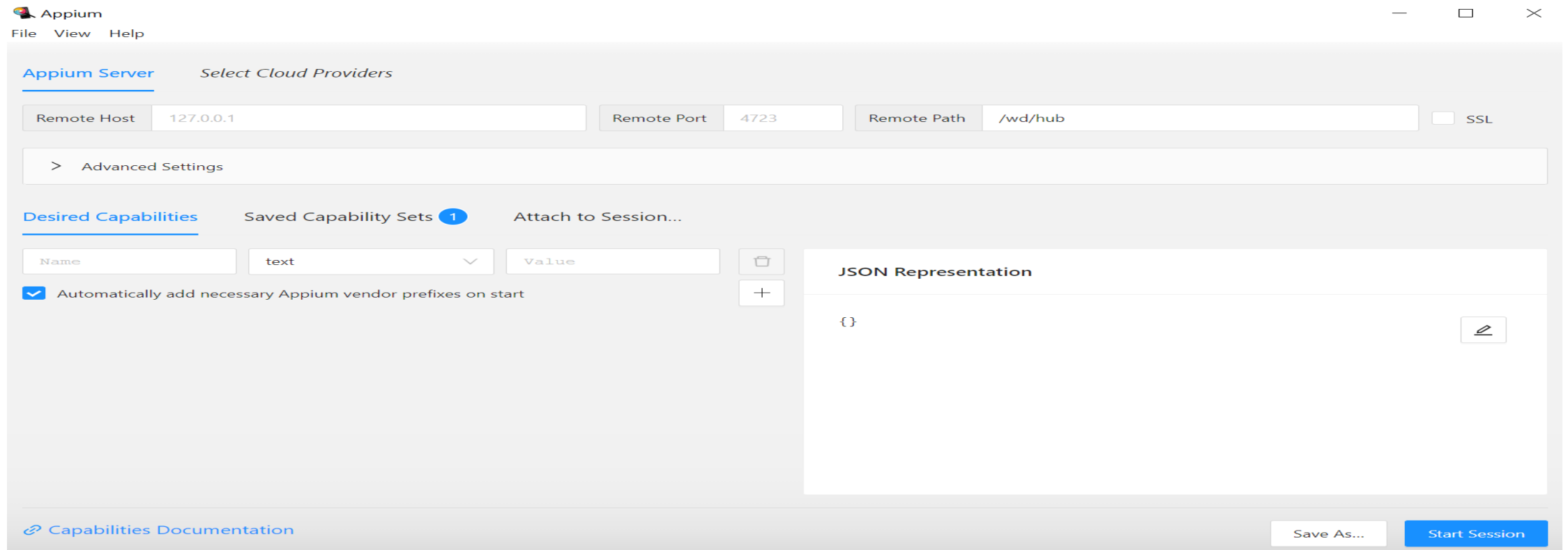
Double click Appium inspector to open and use it to locate elements using various locators.



Mobile App Testing using Emulator

Appium Inspector

- Double click Appium inspector to open and use it to locate elements using various locators.
- Check **Remote Host(127.0.0.1)**, **Remote Port(4723)** and **Remote Path(/wd/hub)** is given properly.



Mobile App Testing using Emulator

Appium Inspector

- Type **platformName** in Desired capabilities Name Field and type **Android** in value Field.
- Once you type it will reflect in JSON Representation as shown below.

Appium Inspector interface showing the 'Desired Capabilities' section.

Appium Server *Select Cloud Providers*

Remote Host: 127.0.0.1 Remote Port: 4723 Remote Path: /wd/hub ☐ SSL

Advanced Settings

☒ Allow Unauthorized Certificates ☐ Use Proxy Proxy URL

Desired Capabilities Saved Capability Sets **1** Attach to Session...

platformName text Android

☒ Automatically add necessary Appium vendor prefixes on start

JSON Representation

```
{  "platformName": "Android"}
```

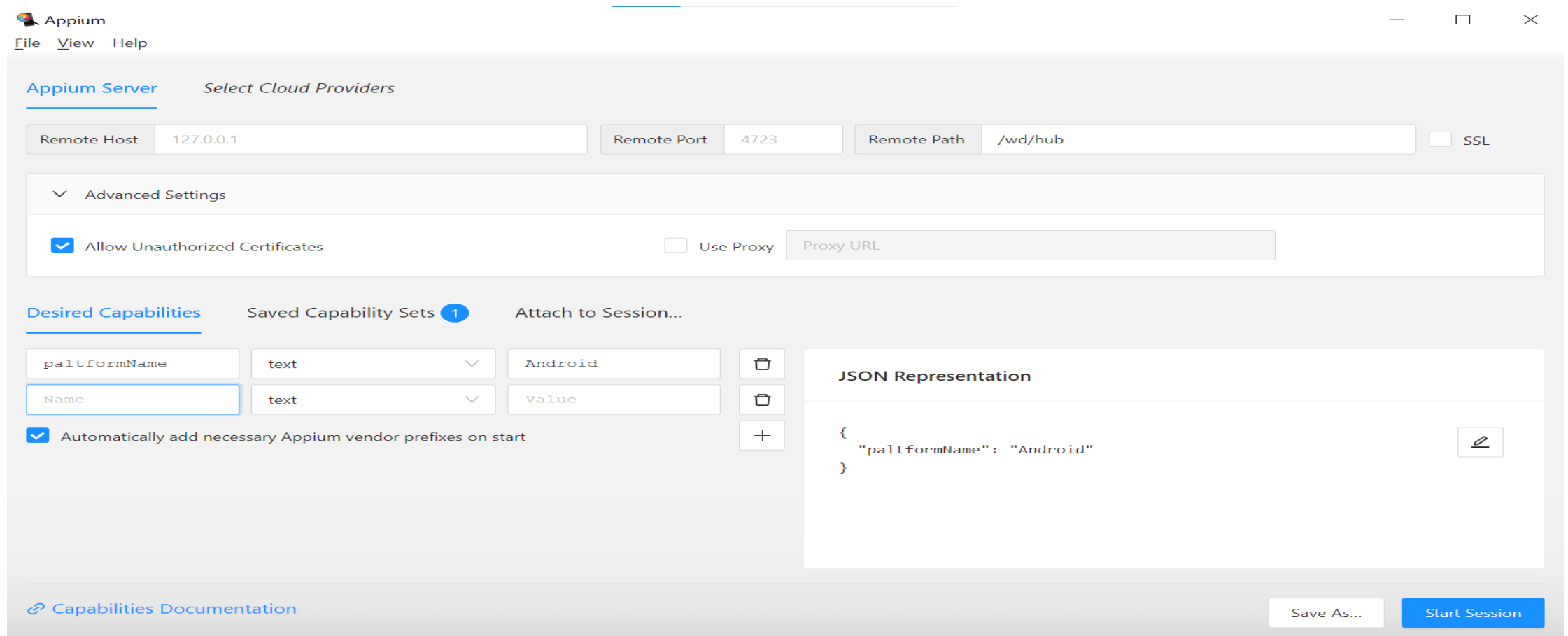
Capabilities Documentation

Save As... Start Session

Mobile App Testing using Emulator

Appium Inspector

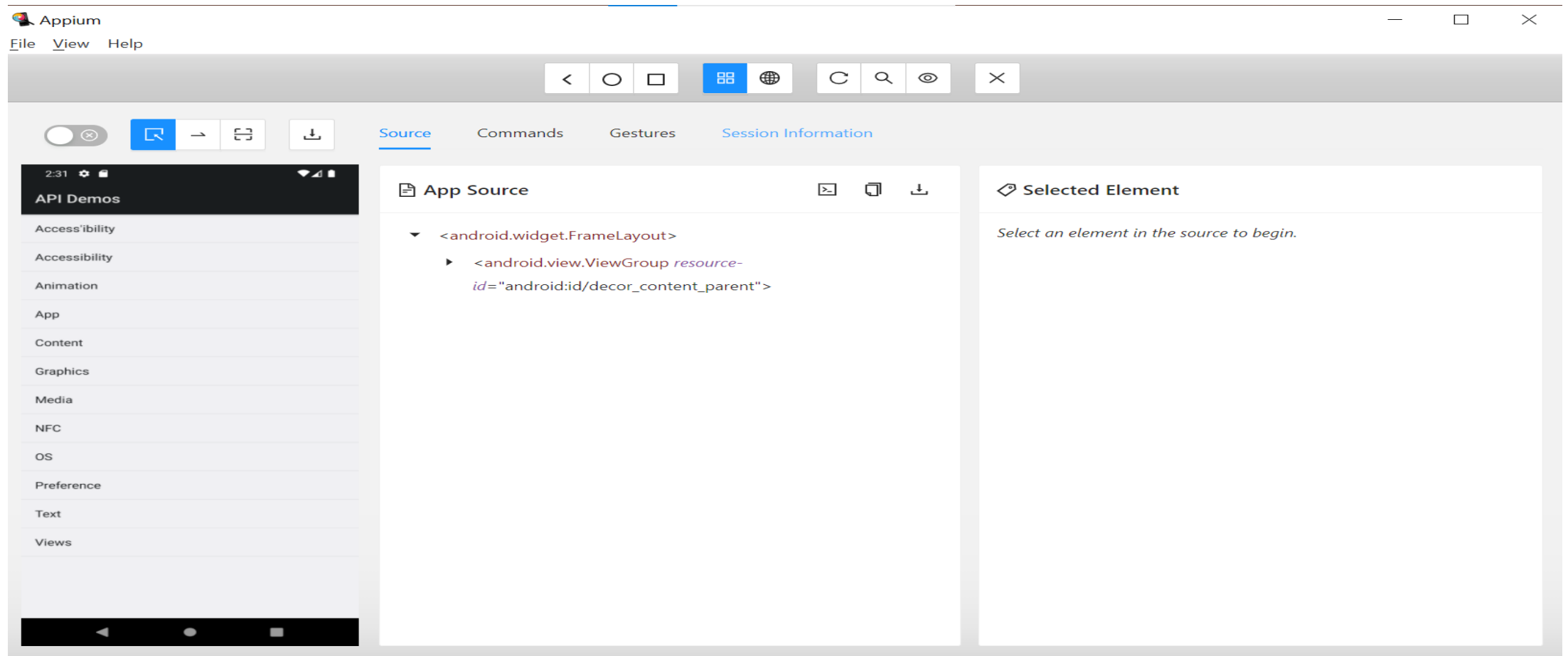
- Then click Start Session



Mobile App Testing using Emulator

Appium Inspector

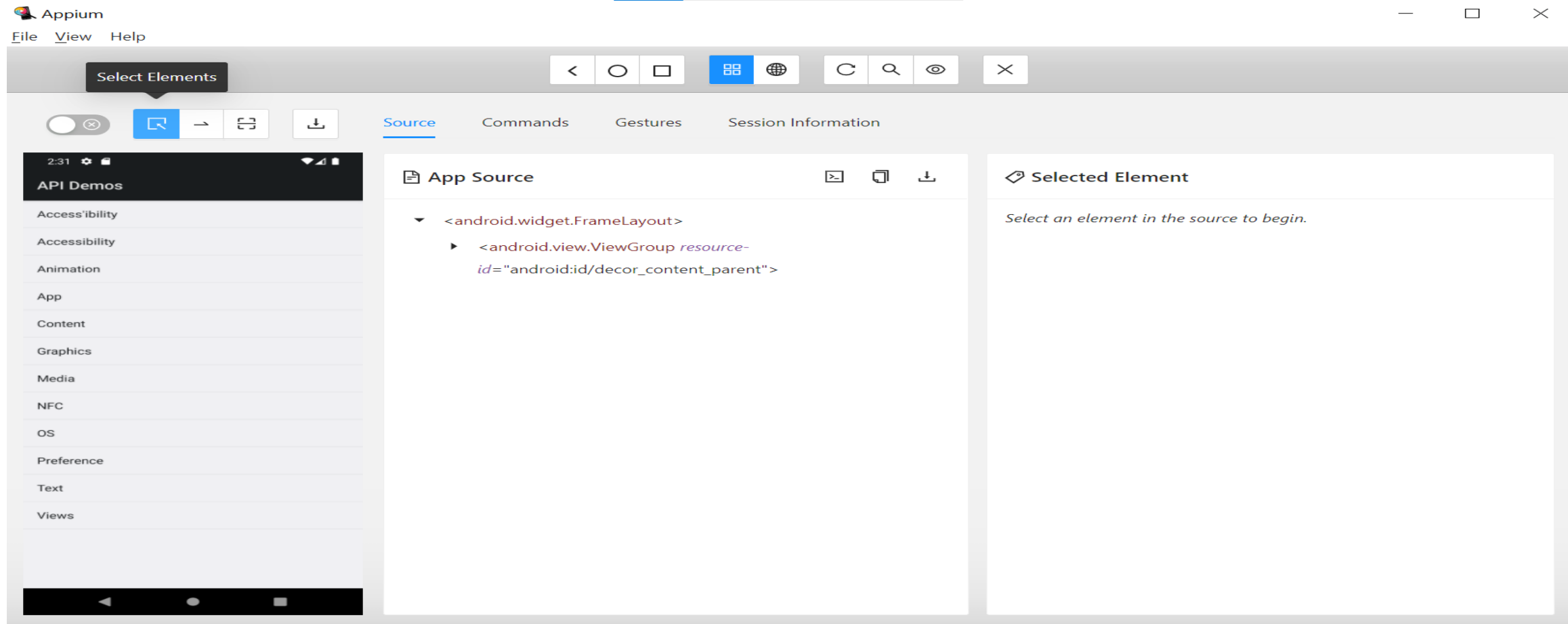
- After successful start you will get the following window.



Mobile App Testing using Emulator

Appium Inspector

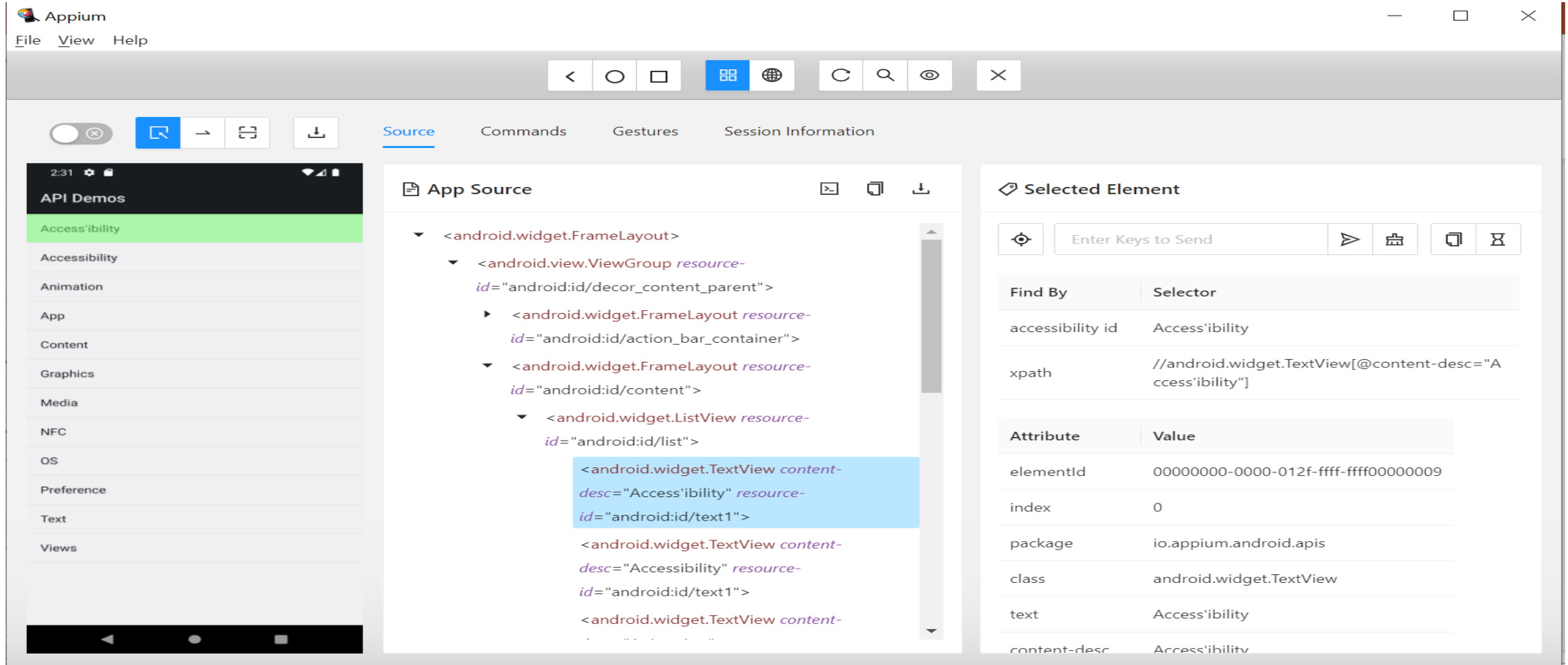
- Click select elements icon and place it over the required elements to the design details.



Mobile App Testing using Emulator

Appium Inspector

- Click select elements icon and place it over the required elements to the design details.



Creating a Mobile App Testing Project and Test scripts

Step 1: Create a Maven Project and follow the project structure

Step 2: Go to pom.xml and add the following dependencies

2.1 Selenium Java

```
<dependency>  
  
<groupId>org.seleniumhq.selenium</groupId>  
  
<artifactId>selenium-java</artifactId>  
  
<version>4.11.0</version>  
  
</dependency>
```

Creating a Mobile App Testing Project and Test scripts

Step 2: Go to pom.xml and add the following dependencies

2.2 io.appium.java-client

```
<!-- https://mvnrepository.com/artifact/io.appium/java-client -->
```

```
<dependency>
```

```
    <groupId>io.appium</groupId>
```

```
    <artifactId>java-client</artifactId>
```

```
    <version>8.5.1</version>
```

```
</dependency>
```

Write the code in the test class as follows

```
public class MobileDemo2 {
    public static AndroidDriver driver; // android driver object creation
    public static void main(String[] args) throws MalformedURLException, InterruptedException {
        // UiAutomator2Options Object creation to set capabilities
        UiAutomator2Options options = new UiAutomator2Options();
        options.setPlatformName("Andriod"); //Platform Name
        options.setCapability("udid", "emulator-5554"); // Device ID
        options.setCapability(MobileCapabilityType.PLATFORM_NAME, MobilePlatform.ANDROID); // to set mobile platform
        options.setCapability("appPackage", "io.appium.android.apis"); // To select the app with package
        options.setCapability("appActivity", "io.appium.android.apis.ApiDemos"); // To select the app with activity
        URL url = new URL("http://127.0.0.1:4723"); // default URL localhost for Appium server
        driver = new AndroidDriver(url,options);
        Thread.sleep(3000);
        WebElement Accessibility1 = driver.findElement(By.id("android:id/text1")); // element located using id(resource-id)
        Accessibility1.click();
        WebElement NP = driver.findElement(By.xpath("//android.widget.TextView[@content-desc='Accessibility Node Provider']")); // element located using xpath with content-desc data
        NP.click();
        driver.navigate().back();
        driver.navigate().back();
    }
}
```

Start Appium Server

Start the Android Virtual Device in Android Studio with required application to test

1. Set up UIAutomator Driver Installation in command prompt:

appium driver install uiautomator2

2. To add wait plugin Installation in command prompt:

appium plugin install --source=npm appium-wait-plugin

3. To use wait plugin When you start appium server use the following command:

appium --use-plugins=element-wait

Mobile App Testing using Emulator

Start Appium Server in Command Prompt and Run the Test Script in Eclipse to check the results.

```
C:\> npm
AppiumDriver@5ae0 Removing session '7b26f17e-e372-473d-ab49-5441ab56b201' from our master session list
debug] [AndroidUiautomator2Driver@e719 (7b26f17e)] Deleting UiAutomator2 session

C:\Users\Arun Kumar>appium --use-plugins=element-wait
Appium] Attempting to load plugin element-wait...
debug] [Appium] Requiring plugin at C:\Users\Arun Kumar\node_modules\appium-wait-plugin
Appium] Welcome to Appium v2.0.0
Appium] Non-default server args:
Appium] {
Appium]   usePlugins: [
Appium]     'element-wait'
Appium]   ]
Appium] }
Appium] Attempting to load driver uiautomator2...
debug] [Appium] Requiring driver at C:\Users\Arun Kumar\node_modules\appium-uiautomator2-driver
Appium] Appium REST http interface listener started on http://0.0.0.0:4723
Appium] You can provide the following URLs in your client code to connect to this server:
Appium]   http://172.21.96.1:4723/
Appium]   http://192.168.210.83:4723/
Appium]   http://127.0.0.1:4723/ (only accessible from the same host)
Appium] Available drivers:
Appium]   - uiautomator2@2.29.5 (automationName 'UiAutomator2')
Appium] Available plugins:
Appium]   - element-wait@2.0.3 (ACTIVE)
--> GET /wd/hub/session/dd34d51b-c97e-4b3d-a1a5-ff71c2116e56/timeouts
{}
debug] No route found for /wd/hub/session/dd34d51b-c97e-4b3d-a1a5-ff71c2116e56/timeouts
<-- GET /wd/hub/session/dd34d51b-c97e-4b3d-a1a5-ff71c2116e56/timeouts 404 28 ms - 262
```