List the Prerequisites of setting up Appium on the system.

Following are the necessary Prerequisites:

1. JDK(Java Development Kit) setup: where appium uses java for execution so it need JDK.
   JDK Version must be 8 or above.
   In the Environmental Variable add JAVA_HOME located to location of JDK under System Variable.
2. npm: For using npm which comes with node.js which is required for installing appium.
   Eg: Installation Steps:
   ```
   npm install appium
   npm install appium-doctor
   appium-doctor –android
   appium
   ```
3. Android Studio and its SDK: Where we can create our own Emulator for android no need to external USB debugged Hardware, Android Studio has the SDK Manager where we can start the emulator. And we can use it in form of file.
   Create a file named emulator2.bat having the following code:
   ```
   @echo off
   cd /d
   “C:\Users\kiran\AppData\Local\Android\Sdk\emulator”
   start emulator -avd setDevice
   Pause
   ```
   In the Environmental Variable -> System variable -> Create a Variable Named ANDROID_HOME set to SDK Location.

4. Appium-doctor: This is to check all the necessary task is checked or not.
5. Real Devices or Emulators/Simulators: To connect the hardware, In your android device,
   Go to -> Settings -> About Phone -> Click on MIUI Version (for Poco F1) multiple times to become a developer -> then go back and go to Additional Settings -> Developer Options -> Turn ON USB Debugging.
6. Eclipse IDE: For user Scripts. Which includes Libraries and Dependencies.

Include both Hardware and Software requirements.

Above are the software requirements and additional software requirements are as fallows:

1. Operation System: Windows
2. Virtual Emulator.

Hardware requirement: given above like Android, Laptop or PC for testing the Mobile based on user scripts.

Explain the Steps to configure Appium for testing on a Android device.

Following are the given steps:

1. Install JDK(Java Development Kit) Version 8 or later:
   Set JAVA_HOME as a system variable setup to path of JDK. in Environmental Variable.
   Importance: Appium uses Java programming language for executing Scripts. Otherwise Appium won’t function as expected.

2. npm(Node Package Manager) with Node.js: For using npm which comes with node.js which is required for installing appium.
   Eg: Installation Steps:
   ```
   npm install appium
   npm install appium-doctor
   appium-doctor –android
   appium
   ```
   Importance: command setup to install appium, appium-doctor, etc.
3. Installing Android SDK: in the Environmental variable create new variable named ANROID_NAME setup to path where sdk is present.
   Android SDK is useful for creating interaction between the Laptop and Android via USB debugging enabled.
4. Appium driver like uiautomator2: Having the interaction way that tests mobile as per tester requirement.
5. Appium Capabilities: In the Script provide the dependencies as a dictionary:
   ```
   desiredDependencies dc = new desiredDependencies();
   dc.setDependency(“key”, “value”);
   {
       deviceName: setDevice,
       udid: emulator-5554,
       platformName: Android,
       platformVersion: 14,
       browserName: Chrome,
       automationName: UiAutomator2,
       chromedriverExecutable: path../chromedriver.exe
   }
   ```

Importance: Setting up the Mobile behavior to Start test based on its Capabilities of DesiredCapabilities.

What is the role of the “Desired Capabilities” and how does it assist testers in creating automation scripts?

Desired Capabilities are the set of key value pairs describing environment of mobile to appium for mobile testing.They are passed from test Scripts to appium server if passed then starts the automation phase.

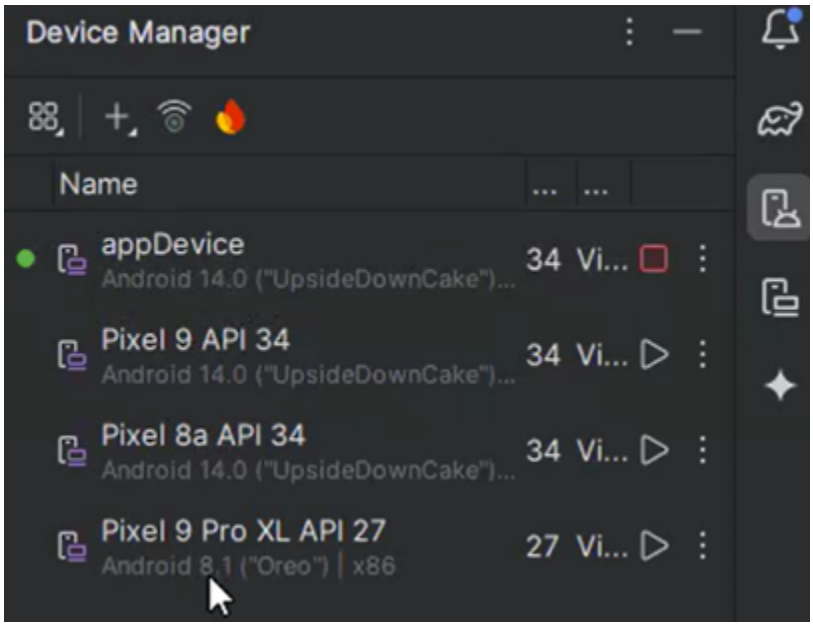

Fig_1: Device manager from Android Studio

Elements in Desired Capabilities:

1. **deviceName:** Its the name provided to the Android device present in Device Manager.
   Eg: key = “deviceName”, value = “appDevice”
   DesiredCapabilities dc = new DesiredCapabilities();
   dc.setCapability(“deviceName”, “appDevice”);
2. **browserName:** set the name of the Browser.
   Eg: dc.setCapability(“browserName”, “chrome”);

3. **Browser Version:** to setup the Browser version.
   Eg: dc.setCapability(“version”, “12.3”);
4. **Platform name and version:** setting up the platform name and version of mobile.
   Eg: dc.setCapability(“platformName”, “Android”);
   dc.setCapability(“platformVersion”, “14”);

Desired Capabilities assisting Testers:

Enabling the tester to change the environments based on testers requirements.

It enables testers to run scripts on different devices, and under different browsers. Improves compactibility checking.

What is the purpose of the Appium Inspector, how does it assist testers in creating automation scripts?

Appium Inspector is primarily used for inspecting the elements of the mobile apps interface to automate the mobile app. It enables testers identifying the element easily due to source code of app is visible.

After Getting the element specifications like id, name, class, xpath etc. QA Testers can use it as an address to trigger elements for automation like pressing, holding, click, sending values etc.

Appium GUI have feature like setting up Capabilities, Inspecting elements, etc.

Describe the process of locating and iterating with UI elements using Appium?

First setting up the Environment:

1. After installing appium start the server.
2. Provide necessary desired capabilities.

For Locating UI Elements:

By ID, XPath, className, name, accessibilityId, UiAutomator etc.

Eg:

```
// by name
WebElement element = driver.findElement(By.name(“”));
// by id
WebElement element = driver.findElement(By.id(“”));
// by xpath
WebElement element = driver.findElement(By.xpath(“”));
// by accessibility id
WebElement element =
driver.findElement(By.accessibilityId(“”));
// by class name
WebElement element =
driver.findElement(By.className(“”));
```

Interacting with UI Element:

```
// For clicking on element
element.click();
// For providing values
element.sendKeys(“values”);
// For clearing the field
element.clear();
// For getting the text
String storeValue = element.getText();
```