**APPIUM PROJECT**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name** | **WSUID** |
| 1 | Surya Kondala Siva Teja Vanga | U579Z927 |
| 2 | Satya Tejaswi Rankireddi | P735X333 |

**Software’s Required:**

* Appium Server
* Appium Inspector
* Android Studio
* Pycharm

We can use either Appium Server or Appium Desktop based on your preferences and requirements for automating Android app testing.

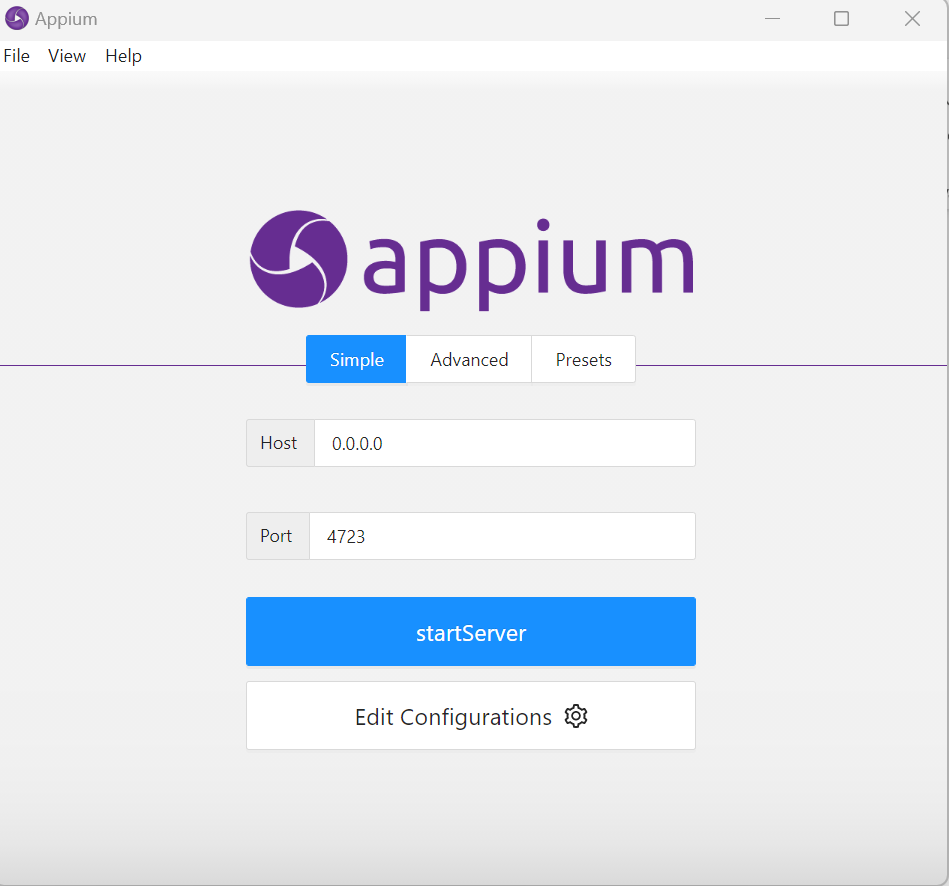
Both Appium Server and Appium Desktop serve the same purpose of automating mobile applications, but they offer different interfaces and features:

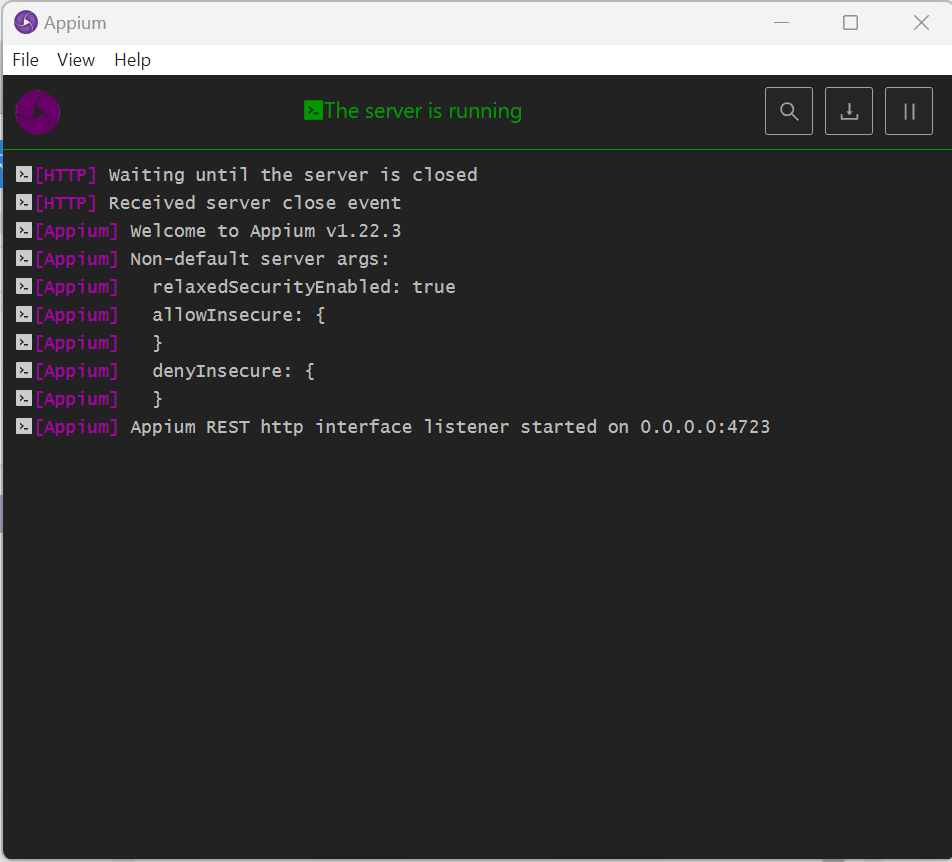
**Appium Desktop:**

Appium Desktop is a graphical user interface (GUI) tool that complements the Appium Server. It simplifies the setup and configuration of the Appium Server, making it easier for testers and developers to automate mobile applications on Android and iOS platforms.

Graphical Interface: Provides a user-friendly interface for interacting with the Appium Server, allowing users to configure settings, manage test sessions, and inspect elements visually.

Element Inspector: Facilitates the examination of the application's user interface hierarchy, aiding in identifying locators and elements for test script development.





A screenshot of a computer program

Description automatically generated

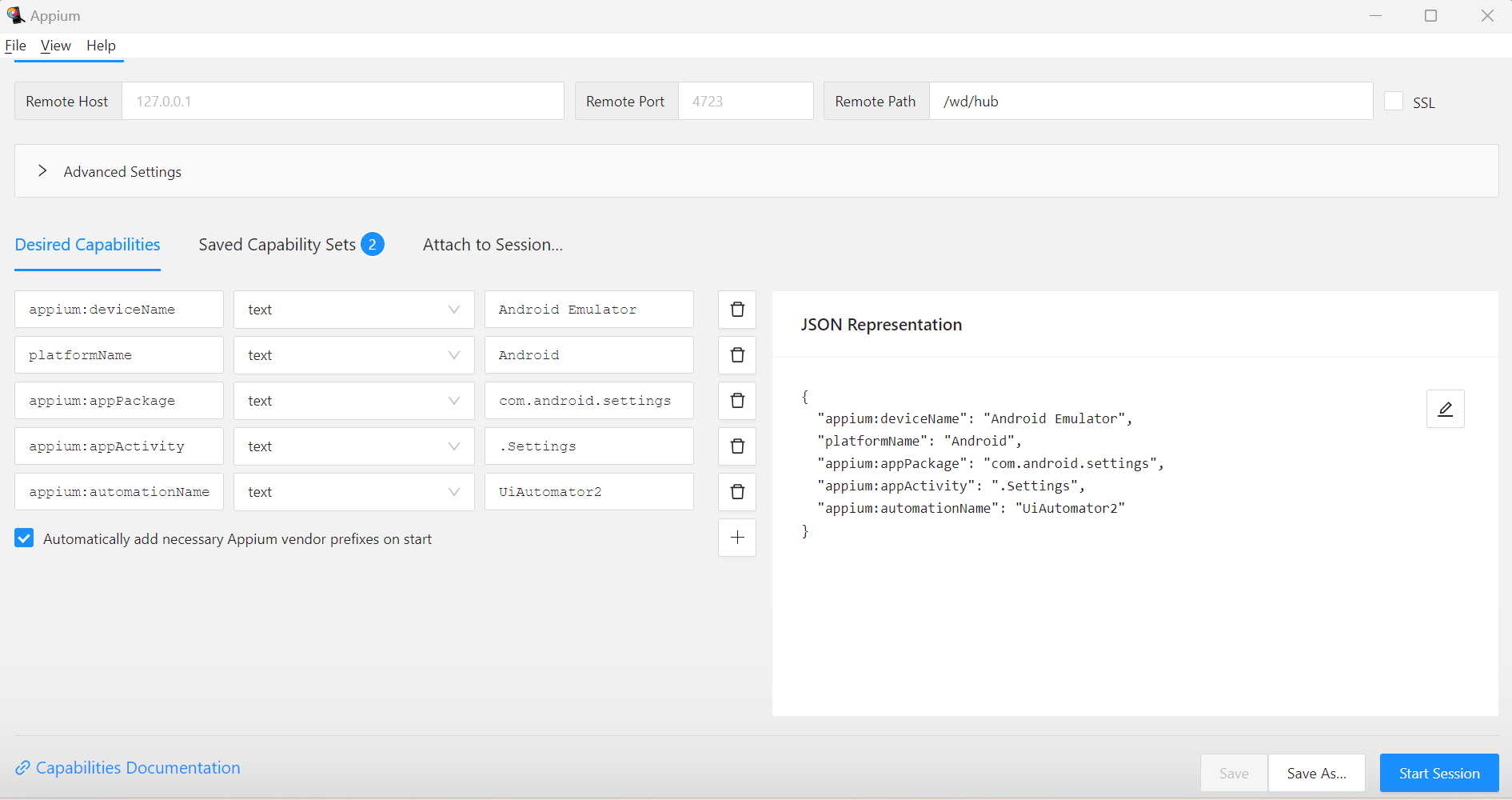
**Appium Inspector:**

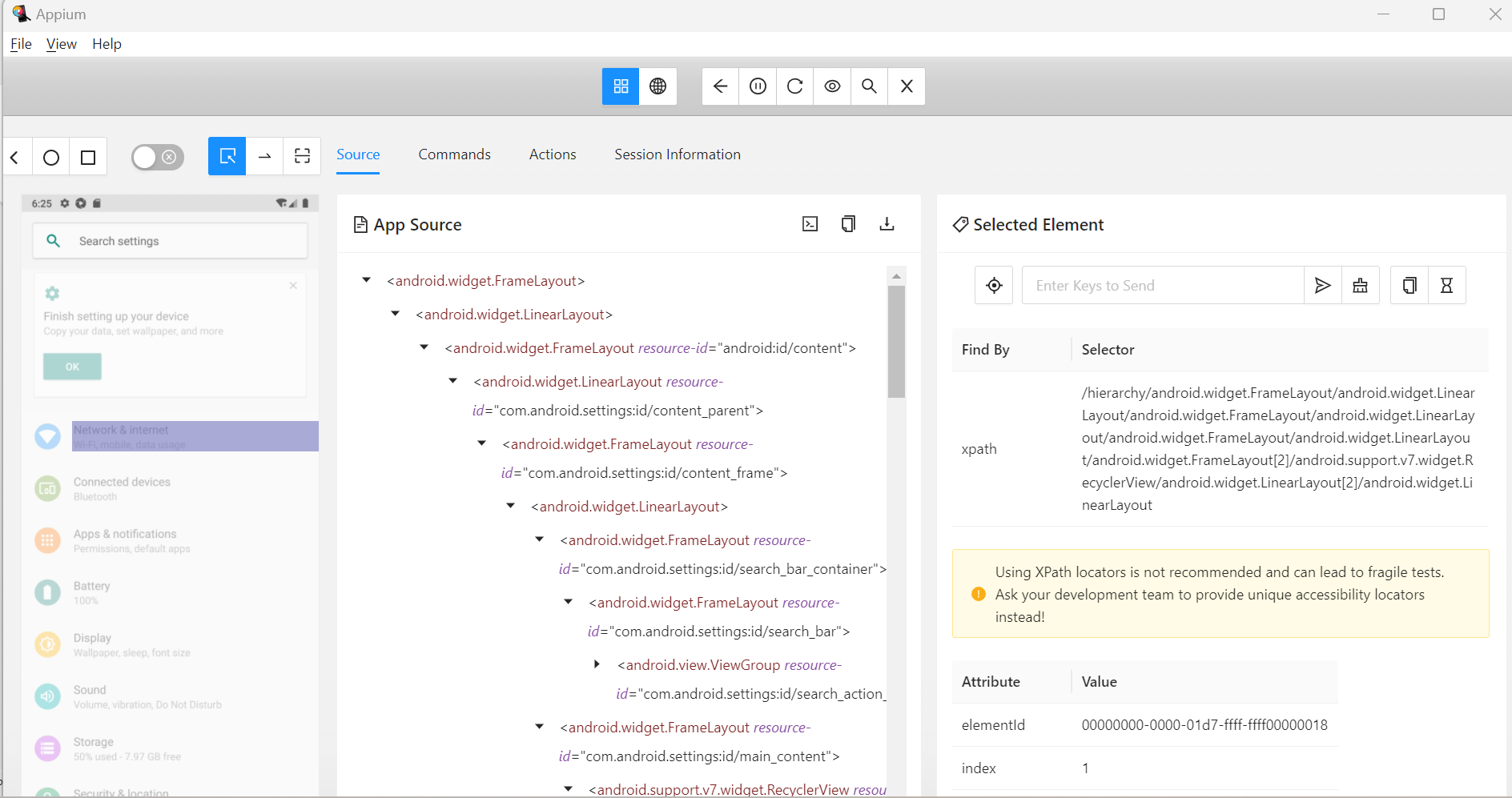
Appium Inspector is a tool integrated into Appium Desktop that aids in inspecting mobile application elements during test automation. It is used to capture element locators and properties, essential for writing effective test scripts.

UI Element Inspection: Allows users to interactively select elements from the mobile application's user interface and view their properties and attributes.

Element Locators: Helps identify unique identifiers (such as resource IDs or Xpaths) required to interact with elements programmatically in test scripts.

Cross-Platform Support: Works for both Android and iOS applications, facilitating cross-platform test automation.





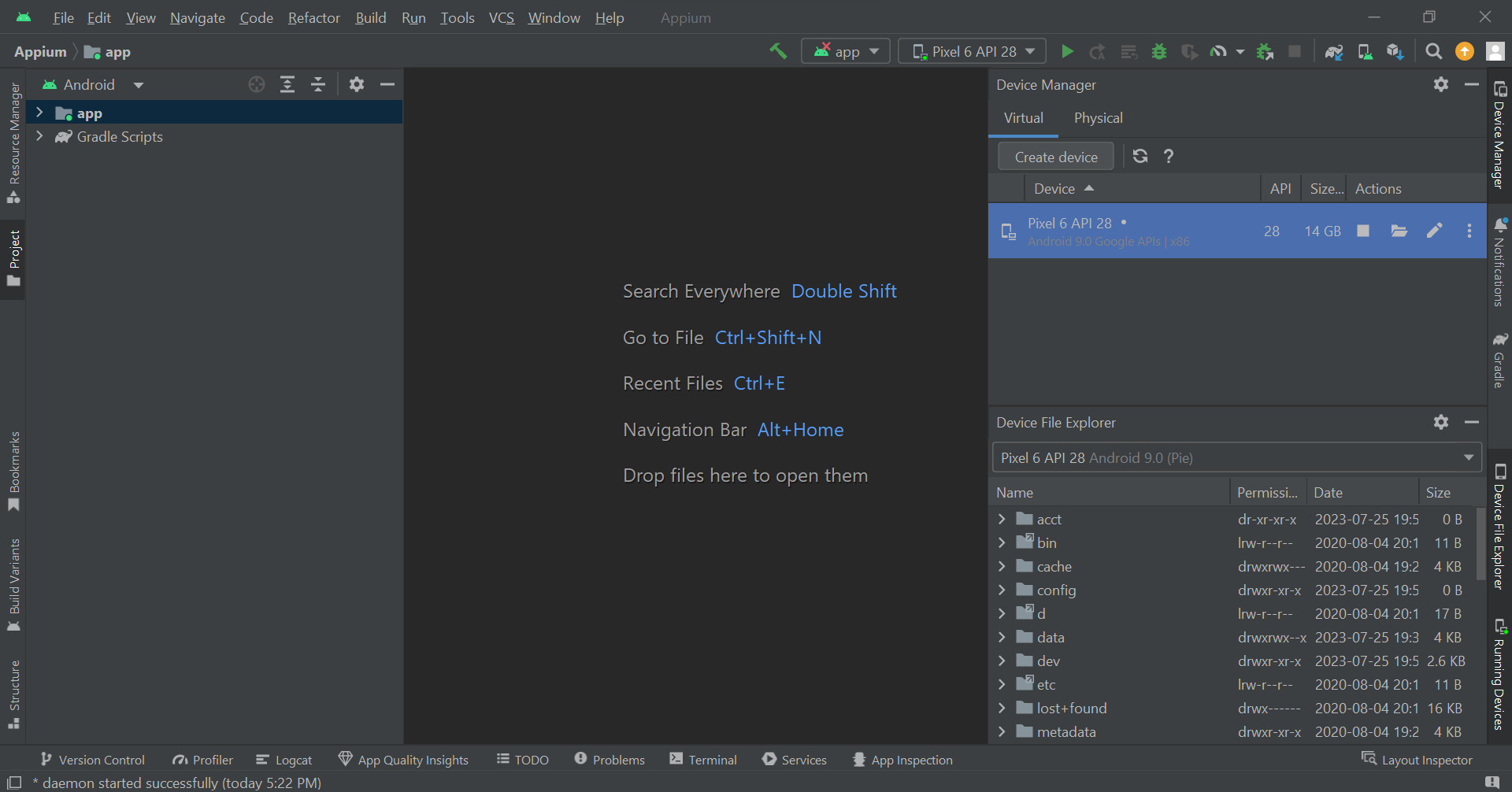
**Android Studio:**

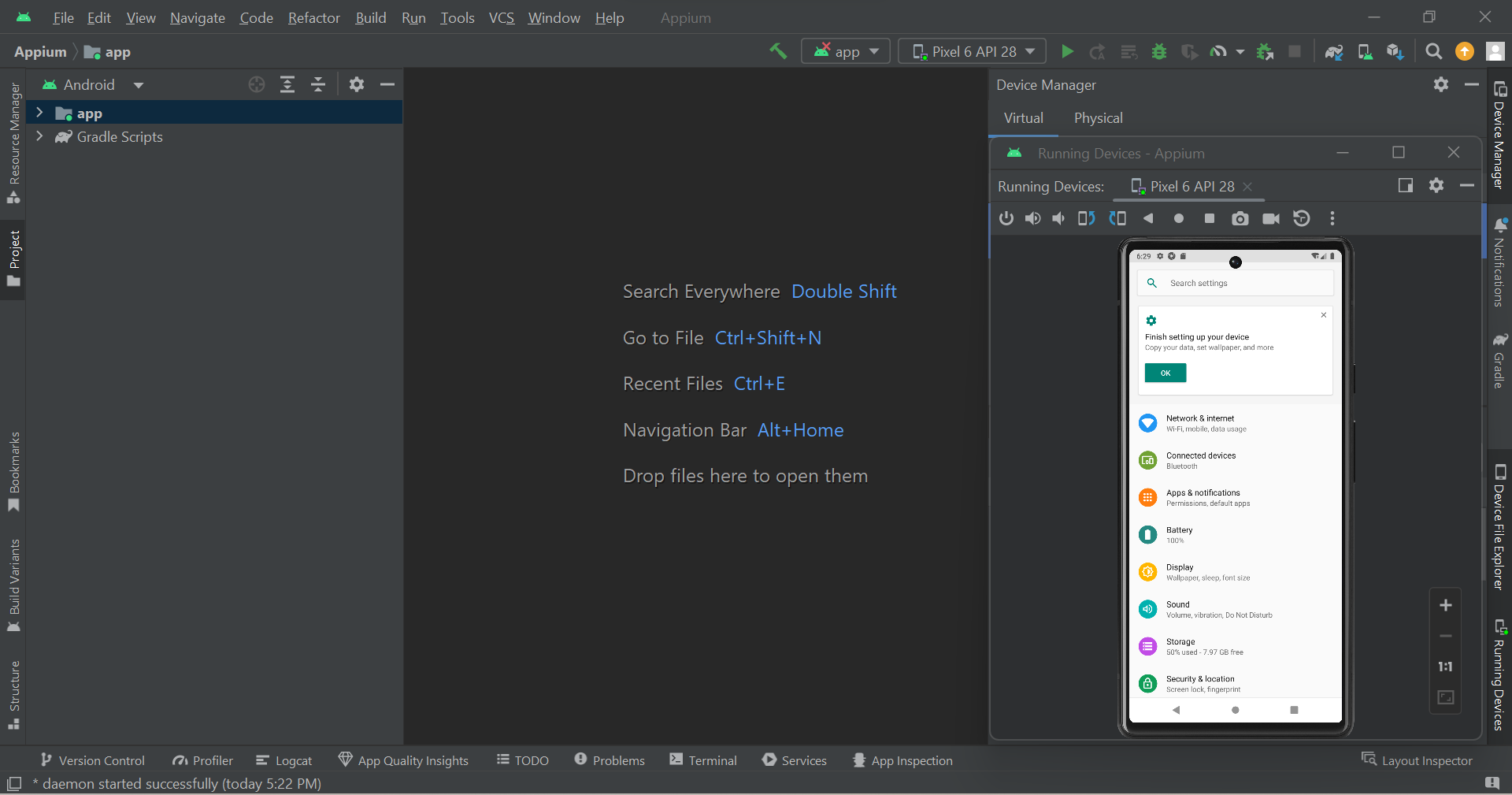
Android Studio is the official Integrated Development Environment (IDE) for Android app development. While not specifically an Appium tool, it is crucial for Android automation as it provides essential tools and emulators for testing Android applications.

Android Virtual Device (AVD) Manager: Enables the creation and management of Android emulators for testing.

Android SDK Manager: Facilitates the installation and management of Android SDKs and required dependencies.

Code Editing and Debugging: Offers a robust code editor and debugging features for Android app development and testing.





A screen shot of a cell phone

Description automatically generated

**PyCharm IDE:**

PyCharm is an Integrated Development Environment (IDE) specifically designed for Python development. It is widely used for scripting and running test automation scripts with Appium.

Python Scripting: Provides a powerful Python code editor with syntax highlighting, autocompletion, and code analysis for writing Appium test scripts.

Test Execution: Allows running test scripts directly from the IDE and provides detailed test execution reports.

Debugging Support: Facilitates debugging of test scripts to identify and fix issues during development.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

This code is a Python script that uses the Appium library to automate various tasks related to Android settings and application management on a connected Android device. It allows you to perform the following actions for a list of applications: installation, activation of device admin permission, deactivation of device admin permission (if enabled), and uninstallation.

**Steps:**

**Main Script Files:**

**AdminAppInstall.py**

**appiumTest.py**

1. **Importing** **Libraries**: The script starts by importing necessary libraries, including time, appium.webdriver, TouchAction from appium.webdriver.common.touch\_action, and various classes from selenium package.

2. **Functions for Different Cases:** Four main functions are defined, each corresponding to a different case of action:

- case1(): Installs the specified APK file on the connected Android device.

- case2(): Activates the device admin permission for the specified application.

- case3(): Deactivates the device admin permission for the specified application (if already enabled).

- case4(): Uninstalls the specified application from the device.

These functions have Appium capabilities and actions to perform the respective tasks.

3. **switch\_case() Function:** This function acts as a switch statement, taking a case number as an argument and executing the corresponding function based on that case number.

4. **applicationsList**: A list containing the names of applications that you want to perform actions on.

5. **Looping through the Applications:** The script iterates through each application in the applicationsList and executes the following actions for each application:

case1: Installs the application using the APK file located in the specified path.

case2: Activates the device admin permission for the application.

case3: Deactivates the device admin permission for the application (if already enabled).

case4: Uninstalls the application from the device.

6. The script prints messages indicating the actions that are being executed or completed successfully. In case there is an exception or any error during the process, corresponding error messages will be printed.