**Appium introduction:**

***Appium is an open source test automation tool used for testing mobile applications.***

It allows users to test various types of mobile applications

(such as Native, Hybrid and Mobile Web) using standard

***WebDriver library.***

With Appium, we can also run automated tests on actual devices like tablets, mobile phones etc, emulators and simulators.

[**An emulator** is a hardware device or software program that enables one computer system (also known as a **host**) to imitate the functions of another computer system (known as the **guest**).

It enables the host system to run software, tools, peripheral devices and other components which are designed for the guest system.

Emulators can be of different types, replicating things such as hardware, software, OS or CPU. However, in most cases hardware architecture is emulated to provide an environment similar to a guest system.

***Emulation techniques are applied to re-create the hardware and software environment of a computer system on a different machine.***

Once the emulator is complete, users can access applications or the OS on the emulated system and the original software can run on the host system. To the users, the experience is the same as if they were using the original guest system.]

[A computer **simulation** or a computer model is a computer program that attempts to simulate an **abstract model** of a particular system. ...

Computer simulations build on, and are a useful adjunct to purely mathematical models in science, technology and entertainment.

]

***Appium is a cross platform tool (code re use between Android and iOS)***

***Supports multiple programming languages.***

**Types of Apps: Web, Native and Hybrid.**

***Native apps***:

A native application is a software program that is developed for use on a ***particular platform or device***.

Because a native app is built for use on a particular device and its OS, it has the ability to use ***device-specific hardware and software.***

*Native applications are developed for a specific platform.*

***Web apps: only on browsers***

***Web applications are developed for multiple platforms.***

***Note:***

***Hybrid mobile application:***

***is a combination of the elements of both Web and***[***Native applications***](https://www.softprodigy.com/mobile-application-development-company/hybrid-native-mobile-app-development)***.***

The advantage of developing hybrid applications is that they are **consistent** and **cross platform UI** that is compatible with most of the devices.

Hybrid app development is less expensive but cannot be used for every type of app.

***Hybrid apps are technically web apps packed in a native app container.***

*Hybrid Apps load in web view.*

***Fundamentals of mobile application Testing:***

1. **Installation**

Verify application is installed successfully.

1. **Incoming call/message handling**

Whenever the application is in use, make sure that any incoming communication doesn’t bother the application.

both the audible and visual notifications are displayed properly on the screen.

The user should be able to take or cancel that call making sure that the application does not hang or crash.

verify whether the user receives proper SMS alerts when the application is running, without harming the app.

1. **Un-Installation**

the mobile application gets un-installed without any error.

1. **Test networking issues**

how the application reacts when there is no network coverage or a network problem arises - the app is capable of displaying ‘network error’ messages under such circumstances.

1. **Application Logo / Name**

they both gets displayed in the device after installation and the user is able to select it.

1. **Splash Screen**  
   When a user selects the application logo, a splash screen should be displayed. Verify whether the splash screen is shown and ensure that the flash is displayed not more than few seconds.
2. **Start / Restart Application**  
   Test whether the start / restart functions are working fine.

Ensure that the time for the application to start / restart is quick.

1. **Application Side Effects**  
   Ensure that the application after installation, doesn’t damage any other application or device.
2. **Exit Application**  
   Test whether the user of the application has options to exit the application by means of all exit modes like End key or Exit option.
3. **Charging / Un-charging**  
   While the application is running and a charger is inserted or removed, then a proper message needs to be displayed saying that the device is running out of battery or needs to be notified that it is charging.
4. **Graphical User Interface**  
   It is also quite vital for testing the application for color schemes, themes, menu styles, font color, font style and much more.
5. **Security**

***Working with Appium: for Functional Test Automation of Apps (iOS + Android)***

***Appium: installation***

***Pre requisite:***

1. ***Xcode***
2. ***Home brew***
3. ***Node***
4. ***Appium***

* npm install -g Appium
* appium -v
* appium

1. ***appium Desktop client***

* ***appium.io – download appium.dmg***

1. ***appium-doctor***

* npm install -g appium-doctor

1. ***JDK***
2. ***Eclipse***
3. ***Android SDK [***brew install android-sdk / brew cask install android-sdk ]

***[Android Studio – an IDE]***

1. ***Platform tools []***
2. ***Appium Client libraries***

***Java client – for advanced activity {from :*** <http://appium.io/downloads.html>***}***

1. ***Selenium Client and WebDriver language bindings***

***Selenium – java (from:*** <https://www.selenium.dev/downloads/>

**Why Appium?**

**Free and open source**

**Supports both iOS and Android**

**No need for app source code**

**Does not re install application**

**Supports multiple frameworks and programming languages**

**Strong and active community.**

**Appium Architecture**

***Appium is an HTTP server written in node.js which creates and handles multiple WebDriver sessions for different platforms like iOS and Android.***

*Appium starts a Test case on the device that spawns a server and listens for proxied commands from main Appium server.*

**Appium follows client Server Architecture.**

**Appium’s philosophy:**

* *No recompilation of App / modify to automate it.*
* *Language independent*
* *No reinvention of wheel to Automate Apps API.*
* *Open source*

**Pros:**

*Complexity is encapsulated.*

*Cross platform*

*No external agent required*

**Cons:**

*Difficult to scale*

*A screenshot of a cell phone

Description automatically generated*

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**