CASE STUDY REPORT
SOFTWARE TOOL:APPIUM

ANAKHA AJKUMAR
Msc CS (DA)
Roll no:07
RAJAGIRI COLLEGE OF SOCIAL SCIENCES

ABSTRACT

Software testing is an important part of the software development life cycle. It makes sure that the software is of good quality by fixing bugs. This can be done with automated testing instead of manual testing to save time and effort. With the help of automated testing tools, the whole testing process can be made to run by itself. This paper is a feasibility study for the testing tool used in industry right now. I did a study on how things work and then briefly talked about the tool I used for the study.

1. INTRODUCTION

Mobile technology has become an important part of daily life. In a short amount of time, more mobile apps are being made. Most people now use their phones' browsers to open web pages instead of their computers. It is important to make sure that our web application works with mobile browsers and apps.

The developer of an app needs to make sure that it works well on many different platforms quickly. Testing is a key part of the software development life cycle because it improves the system's quality, reliability, and performance. Mobile application testing could be done by hand, and the results were checked by looking at the screen. However, it takes a lot of time. Automation is the main thing that makes testing easier and takes less time. Automation testing uses software to run tests and compare the actual results with what was expected. Automated testing helps make software that is good, strong, and reliable. It helps a software development company in many ways, like making testing more efficient, effective, and on-time. Mobile automation testing is a very good way to test these kinds of apps. Automation of software testing is the new way to make sure an application works well and quickly. The Appium tool helps to make features that are strong.With the Appium tool, you can automate testing for native, hybrid, and web mobile apps. So, there needs to be a framework that can be used to test different browsers and apps for mobile devices. In this paper, a new automated testing framework for mobile applications is suggested. Using the Appium tool, the new framework will let the tester do automated testing of different browsers and apps on mobile devices.

2.WORKING

Appium is a test automation tool for both native and hybrid mobile apps that is free to use. It lets testers write automated tests in different programming languages, like Java, Python, and Ruby, and run them on different mobile platforms, like iOS and Android.Appium talks to mobile devices and simulators/emulators using the WebDriver protocol. This lets you write automated tests that simulate how a user would interact with the app. Appium also gives you access to functions that are unique to your device, like the GPS, camera, and contacts.

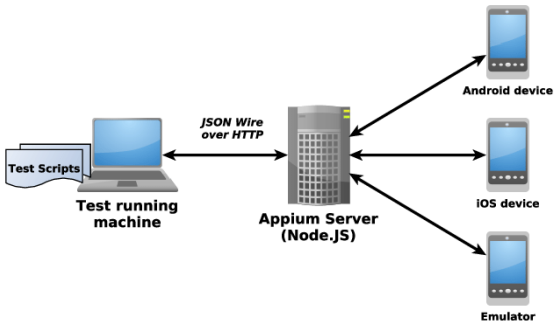
Appium works with a number of automation frameworks, like TestNG, JUnit, and Cucumber, so it can be used with different testing and development processes. It also has a lot of documentation and a big group of users and people who add to it.Because of its flexibility, portability, and open-source nature, Appium is a popular solution for mobile application testing. It is suitable for functional, regression, and acceptance testing, as well as pipelines for continuous integration and delivery (CI/CD).

❖ Architecture of Appium

Appium is an HTTP server written in Node.js that has a REST (Representational State Transfer) API and supports Selenium WebDriver. It operates on a client/server model. Appium supports firing tests using the provided WebDriver client.

The REST API perform the following action:

- Receive the connection from the client side
- Listens to the client side
- Executes the command on a mobile device
- Returns the command execution status as an HTTP response to the client



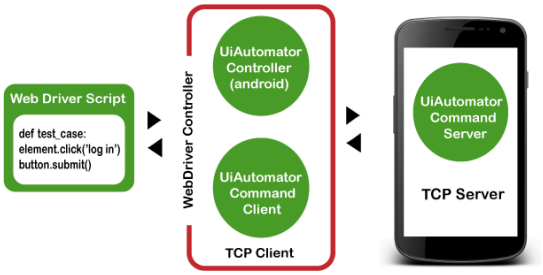
Appium automatically interacts with an application by exploiting the behaviour of various UI components such as buttons, text boxes, and links. It may be employed to develop and execute tests against the provided application several times over different sessions.

1. When we install Appium, we also install a server on our PC that exposes the REST API.
2. It receives client command and connection requests and executes them on devices such as iOS or Android.

❖ Appium in Android

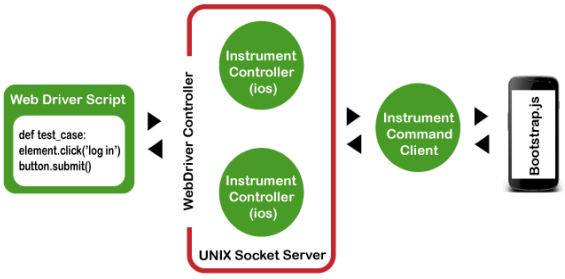
On Android, Appium sends the command to a script running on the device that uses UIAutomator. UIAutomator is an Android framework for automating the user interface that lets you run Junit test cases on the device directly from the command line. Even though it is written in Java, Appium lets it be run from any language that supports WebDriver.

Bootstrap.jar is used by Android, and it acts as a TCP server. Using UIAutomator, it is used to send the test commands that tell the Android device what to do.

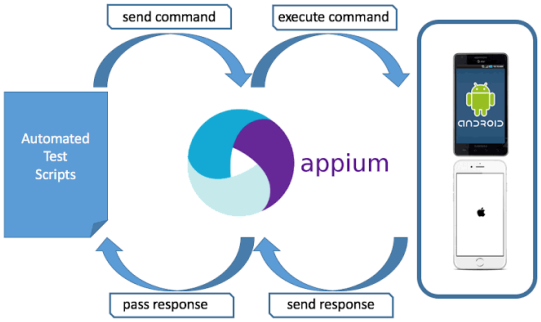


❖ Appium in iOS

As UIAutomator is used by Android, UIAutomation is used by iOS. Appium sends the command to a UIAutomation test case running on the Mac instruments environment, just like Android does. Apple gives us this "instrument" for apps that lets us build, profile, and control iOS apps, among other things. On the other hand, it also has a part for automating tasks, where you can write JavaScript commands. It talks to the Application UI using the UIAutomation API. The same libraries are used by Appium to automate iOS apps.



In general,Appium works by using the WebDriver protocol to connect the mobile device or emulator to the test automation code. This connection is made possible by the Appium server. Test automation code sends commands to the Appium server, which then turns those commands into actions on the device, like tapping, scrolling, and typing text. In response to these actions, the device sends feedback to the Appium server, which then sends it back to the automation code to be checked. This lets testers find bugs or other problems in the app. Appium works with many different mobile platforms, programming languages, and testing frameworks. This makes it a flexible and powerful tool for testing mobile apps.



One of the main reasons Appium is so popular is that it is a lot like Selenium, which has been the most popular open-source test automation framework for more than a decade. Appium is similar to Selenium in that it uses the Webdriver protocol and lets you write tests in many different languages. So, teams that already use Selenium can easily use Appium, as long as they stick to the same test abstractions and frameworks.

Also, because Appium is designed to use a black box testing method, its architecture doesn't need access to the source code of the app being tested. Instead, it only needs access to the app's binary files (APK for Android, and IPA for iOS). So, it's a natural fit for QA teams that get builds from developers and can jump right into automation without having to set up a dev environment or get secure access to it. This also lets testers check a version of the product that is much closer to the one that is

shipped to users. This reduces the chance that something will work in the development build but not in the production build.

3. Conclusion

In conclusion, Appium is a useful tool for automating tests of mobile apps. It's a good choice for testers and developers because it's open source, flexible, and works with different programming languages and mobile platforms. By using the WebDriver protocol to connect the mobile device or simulator to the test automation code, Appium lets testers write and run automated tests that simulate how a user would interact with the app. Because Appium can find problems and bugs in an app, testers can make sure that the app gives users a good experience. Even though there may be problems with using Appium to test mobile apps, these problems can be solved by using best practises like prioritising device coverage, keeping test scripts updated, and managing test data. Overall, Appium is a powerful and flexible tool that can help make sure mobile apps are good and reliable.

4.REFERENCES

- **https://www.javatpoint.com/appium**
- **https://www.moquality.com/blog/Appium-The-Ultimate-Guide**
- **https://www.browserstack.com/guide/appium-tutorial-for-testing**