An

Seminar Report on

# Appium- An Automation Testing Tool

Submitted as a partial fulfilment of

DIPLOMA IN INFORMATION TECHNOLOGY

Submitted By

Mr. Jadhav Manish Shashikant (2201933)

PRN No. - 2030408246006

DEPARTMENT OF INFORMATION TECHNOLOGY

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY'S

INSTITUTE OF PETROCHEMICAL ENGINEERING

Lonere, Tal. Mangaon, Dist. Raigad, Maharashtra-402103

Academic Year: 2022-2023

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**INSTITUTE OF PETROCHEMICAL ENGINEERING**

Lonere, Tal. Mangaon, Dist. Raigad, Maharashtra-402103

# CERTIFICATE

This is to certify that the Seminar report entitled

**Appium- An Automation Testing Tool**

Submitted by

**Mr. Jadhav Manish Shashikant (2201933)**

**PRN No. - 2030408246006**

is a bonafide work of student of final year Diploma in Information Technology submitted in partial fulfilment for the award of Diploma in Information Technology as prescribed by Dr. Babasaheb Ambedkar Technological University's, Institute of Petrochemical Engineering, Lonere during the academic year 2022-2023.

| | |
|---|---|
| **Prof. S. S. Kamble** | **Prof. K. R. Korpe** |
| **Internal Guide** | **Seminar Coordinator** |

**Prof. S. M. Gaikwad**

**Head of Department**

**Place: Lonere**

**Date:**

# Acknowledgement

It is indeed a matter of great pleasure & privilege to be able to present this Seminar on **"Appium- An Automation Testing Tool"** under the valuable guidance of **Prof. S. S. Kamble mam**, thanks for her valuable guidance, advice and constant aspiration to my work.

I have made this report file on the topic Appium; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic. My efforts and wholehearted co-corporation of each and every one has ended on a successful note. I express my sincere gratitude to **Prof. S. S. Kamble mam**, who assisting me throughout the preparation of this topic. I thank her for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it

I would also like to thank to Project Co-ordinator **Prof. K. R. Korpe sir,** my Institute as well as department for giving the opportunity to present this seminar, it will indeed help me to improve my speaking skills and I will be more confident while presenting for the industrial purpose after completion of the studies.

Thank you!

**Mr. Jadhav Manish Shashikant**

Department of Information Technology

Dr. Babasaheb Ambedkar Technological University's

Institute of Petrochemical Engineering, Lonere

# ABSTRACT

Due to the extensive amount of benefits that the application bring including easy accessibility, enhanced user engagement and retention, there has been a drastic shift towards the usage of Applications. Given the increased use of the Applications, testing those apps has become even more challenging in terms of covering an exhaustive list of models. This paper outlines the basic strategies and structure for a successful Automation Testing using Appium. It highlights the common mistakes that are being made while automating the testing process. It also defines the key best practices to be followed to have an efficient app test automation using Appium.

# INDEX

# 1.Introduction to Appium

Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms. **Native apps** are those written using the iOS, Android, or Windows SDKs. **Mobile web apps** are web apps accessed using a mobile browser (Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android). **Hybrid apps** have a wrapper around a "I" – a native control that enables interaction with web content. Projects like Apache Cordova make it easy to build apps using web technologies that are then bundled into a native wrapper, creating a hybrid app.

Importantly, Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API. This enables code reuse between iOS, Android, and Windows test suites.

The success of Appium as a tool is a testament to the huge demand for Appium experts in the industry.

## Appium Philosophy:-

Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

1. You shouldn't have to recompile your app or modify it in any way in order to automate it.
2. You shouldn't be locked into a specific language or framework to write and run your tests.
3. A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.
4. A mobile automation framework should be open source, in spirit and practice as well as in name.

## Appium Design:-

So how does the structure of the Appium project live out this philosophy? We meet requirement #1 by using vendor-provided automation frameworks under the hood. That way, we don't need to compile in any Appium-specific or third-party code or frameworks to your app. This means **you're testing the same app you're shipping**. The vendor-provided frameworks we use are:

- iOS 9.3 and above: Apple's XCUITest
- iOS 9.3 and lower: Apple's UIAutomation
- Android 4.3+: Google's UiAutomator/UiAutomator2
- Windows: Microsoft's WinAppDriver

We meet requirement #2 by wrapping the vendor-provided frameworks in one API, the WebDriver API. WebDriver (aka "Selenium WebDriver") specifies a client-server protocol (known as the JSON Wire Protocol). Given this client-server architecture, a client written in any language can be used to send the appropriate HTTP requests to the server. There are already clients written in every popular programming language. This also means that you're free to use whatever test runner and test framework you want; the client libraries are simply HTTP clients and can be mixed into your code any way you please. In other words, Appium & WebDriver clients are not technically "test frameworks" – they are "automation libraries". You can manage your test environment any way you like!

We meet requirement #3 in the same way: WebDriver has become the de facto standard for automating web browsers, and is a W3C Working Draft. Why do something totally different for mobile? Instead we have extended the protocol with extra API methods useful for mobile automation.

## Appium Concepts:-

### Client/Server Architecture:

Appium is at its heart a webserver that exposes a REST API. It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response representing the result of the command execution. The fact that we have a client/server architecture opens up a lot of possibilities: we can write our test code in any language that has a http client API, but it is easier to use one of the Appium client libraries. We can put the server on a different machine than our tests are running on. We can write test code and rely on cloud services to receive and interpret the commands.

## Session:

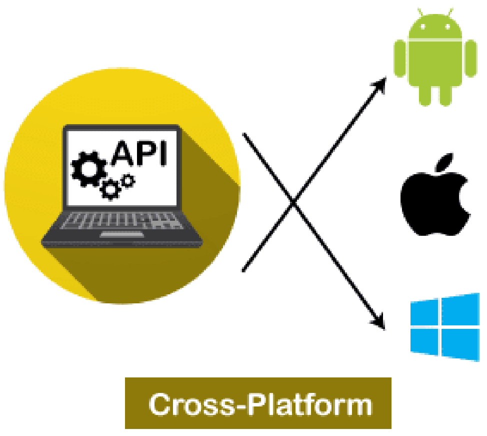
Automation is always performed in the context of a session. Clients initiate a session with a server in ways specific to each library, but they all end up sending a POST /session request to the server, with a JSON object called the 'desired capabilities' object. At this point the server will start up the automation session and respond with a session ID which is used for sending further commands.

## Desired Capabilities:

Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session we're interested in starting up. There are also various capabilities which can modify the behavior of the server during automation. For example, we might set the platformName capability to iOS to tell Appium that we want an iOS session, rather than an Android or Windows one. Or we might set the safariAllowPopups capability to true in order to ensure that, during a Safari automation session, we're allowed to use JavaScript to open up new windows. See the capabilities doc for the complete list of capabilities available for Appium.

# 2.What is Appium?

**Appium** is an open-source automation mobile testing tool, which is used to test the application. It is developed and supported by **Sauce Labs** to automate native and hybrid mobile apps. It is a cross-platform mobile automation tool, which means that it allows the same test to be run on multiple platforms. Multiple devices can be easily tested by Appium in parallel.



In today's development area, the demand for mobile applications is high. Currently, people are converting their websites into mobile apps. Therefore, it is very important to know about mobile software automation testing technology and also stay connected with new technology. **Appium** is a mobile application testing tool that is currently trending in **Mobile Automation Testing Technology**.

Appium is used for automated testing of **native**, **hybrid**, and **web** applications. It supports automation test on the simulators (iOS) and emulators (Android) as well as physical devices (Android and iOS both). Previously, this tool mainly focused on IOS and Android applications that were limited to mobile application testing only. Few updates back, Appium declared that it would now support desktop application testing for windows as well.

Appium is very much similar to the **Selenium Webdriver** testing tool. So, if you already know Selenium Webdriver, Appium becomes very easy to learn. Appium has **NO dependency** on mobile device OS because it has a framework that converts the Selenium Webdriver commands to UIAutomator and UIAutomation commands for Android and iOS respectively, that depends on the device type rather than the OS type.

It supports several languages such as Java, PHP, Objective C, C#, Python, JavaScript with node.js, and Ruby, and many more that have Selenium client libraries. Selenium is the backend of Appium that provides control over the functionality of Selenium for testing needs.

## Features of Appium

- Appium does not require application source code or library.
- Appium provides a strong and active community.
- Appium has multi-platform support i.e., it can run the same test cases on multiple platforms.
- Appium allows the parallel execution of test scripts.
- In Appium, a small change does not require re-installation of the application.
- Appium supports various languages like C#, Python, Java, Ruby, PHP, JavaScript with node.js, and many others that have Selenium client library.

### Appium Doctor:

Attempts to diagnose and fix common Node, iOS and Android configuration issues before starting Appium.

# 3.Appium Servers, Clients, Desktop

## Appium Server

Appium is a server written in Node.js. It can be built and installed from source or installed directly from NPM:

```
$ npm install -g 11ppium

$ 11ppium
```

The beta of Appium is available via NPM with npm install -g appium@beta. It is the development version so it might have breaking changes. Please uninstall appium@beta (npm uninstall -g appium@beta) before installing new versions in order to have a clean set of dependencies.

## Appium Clients:

There are client libraries (in Java, Ruby, Python, PHP, JavaScript, and C#) which support Appium's extensions to the WebDriver protocol. When using Appium, you want to use these client libraries instead of your regular WebDriver client.

## Appium Desktop:

There is a GUI wrapper around the Appium server that can be downloaded for any platform. It comes bundled with everything required to run the Appium server, so you don't need to worry about Node. It also comes with an Inspector, which enables you to check out the hierarchy of your app. This can come in handy when writing tests.

## 4.Appium Invention

- Appium was originally developed by Dan Cuellar in 2011 under the name of “iOSAuto”, written in the C# programming language.

- The Program was open-sourced in August 2012 using the Apache 2 license.
- In January 2013, Sauce Labs agreed to fund Appium’s development and motivated its code to be rewritten using Node.js.

- Appium won the 2014 *Bossie award* of InfoWorld for the best open source desktop and mobile software.
- Appium was also selected as an *Open Source Rookie of the Year* by Black Duck Software.
- In October 2016, Appium joined the JS Foundation.
- Initially as a mentor program, it graduated in August 2017.

# 5.Manual Testing vs Automated Testing

Software testing can be classified into two categories mainly as manual testing, and automated testing and can be distinguished as follows:

In manual testing, no programming can be done to write sophisticated tests which fetch hidden errors and points to missing information while in automation testing testers can program sophisticated tests to bring out hidden errors and points to missing information.

Secondly, manual testing requires huge investment in human resources but is also less reliable as tests may not be performed with precision each time because of human errors. This type of testing becomes slow, tedious and time consuming. With the advent of increasing number of mobile applications, Automation provides us with a solution , by performing same operation each time they are run eliminating risk of errors as this process makes use of scripts and allowing us to test more number of possibilities to reduce further errors.

Katja Karhu summarizes the difference between the two categories by suggesting that automated software testing should be used to prevent new errors in the already tested working modules, while manual testing is better used for finding new and unexpected errors (K. Karhu, T. Repo and K. Smolander, 2009). The two approaches are complementary to each other, automated testing can perform a large number of test cases in little time, whereas manual testing uses the knowledge of the tester to target testing to the parts of the system that are assumed to be more error-prone.

# 6.Why Choose Appium?

Appium became the major player in test automation landscape due to following reasons.

- **Open Source:**

  Not only Appium is free of cost, but also open source. This means Appium can be tweaked infinitely to work according to your whims.

- **Cross-Platform:**

  Appium can test any mobile application whether it be a native one, a hybrid one or even a web app. Whether the app works only on iOS or Android or even both. Appium has you covered. Above that, Appium itself can be installed on Windows, Mac and Linux.

- **No Source Code:**

  Automation testing application till date would require access to the application code libraries and the source code. Appium brings something new to the table by completely disregarding this fundamental aspect of automation testing.

- **No reinstallation:**

  Appium's philosophy believes in testing an application without reinstalling it and modifying it in any way.

- **Highly Flexible:**

  It supports multiple scripting languages like Java, Python, C#, Javascript etc. that makes it easy for the user to select a scripting language based on their convenience and work across multiple platforms. Also testing the native apps using Appium doesn't require any SDK or app recompiling. In fact, in most of the cases, it doesn't even require any code change to work on Android and iOS.

- **Easy to learn:**

  As Appium is built on Selenium, it doesn't require any time for the Selenium engineers to ramp up on the tool. Apparently, Appium is a wrapper that translates Selenium commands into iOS and Android commands to interact with the elements of the application under test. Furthermore, all Selenium functionality is available in Appium.

- **High Community Support:**

  Another major benefit which Appium brings in is the large community of contributors available on all the major networking portals and are striving to keep the users updated on the latest trends on the tool.

# 7.Types of Application

Appium has the ability to deal with all sorts of applications, i.e., native, hybrid, and web.

## 1.Native Applications:-

Applications that are developed, keeping a certain platform in mind are called native applications. Native applications are developed using specific software development kits and tend to be performance focused. Native apps are generally the way to go if the development budget is not an issue.

Some advantages of native applications are as follows –

- Excellent performance
- They generally tend to look refined and polished
- Native applications seamlessly integrate with device hardware
- Most native applications have an intuitive UI/UX

Some examples of native applications include Spotify, Snapchat, Pintrest, Skype.

## 2.Web Application:-

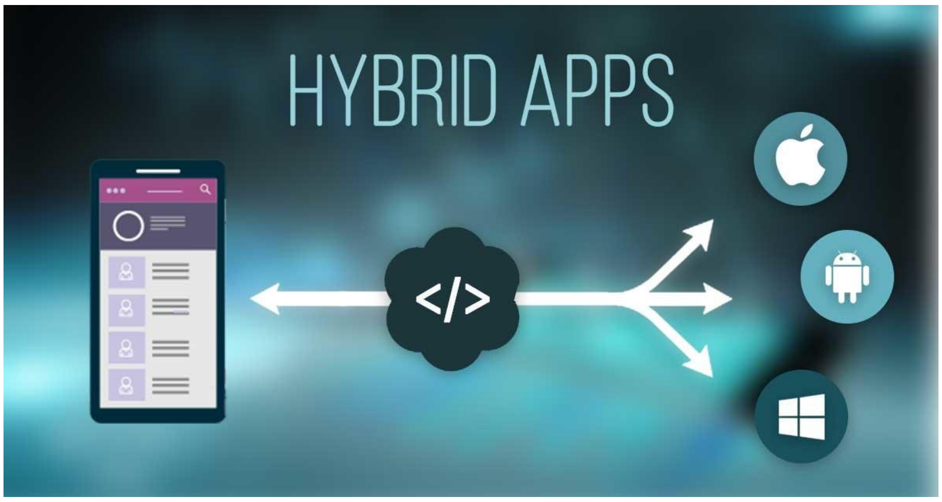
These kinds of applications used to be really famous until the concept of native applications came along. Web applications as the name suggests, run on a browser. Since they run on a browser, they generally don't require any sort of complicated installation and the development process is cheap from a budget perspective too. Since the application is not developed for a particular platform, web-based development languages like HTML/CSS/JavaScript are used for their development.

Below is a list of advantages of web applications –

- Lower development cost
- Easier to maintain
- Web-apps don't need to be approved by application marketplaces
- No installation or user updates

Examples of successful web applications include Ali Express, Flipkart Lite and Washington Post.

## 3.Hybrid Application:-



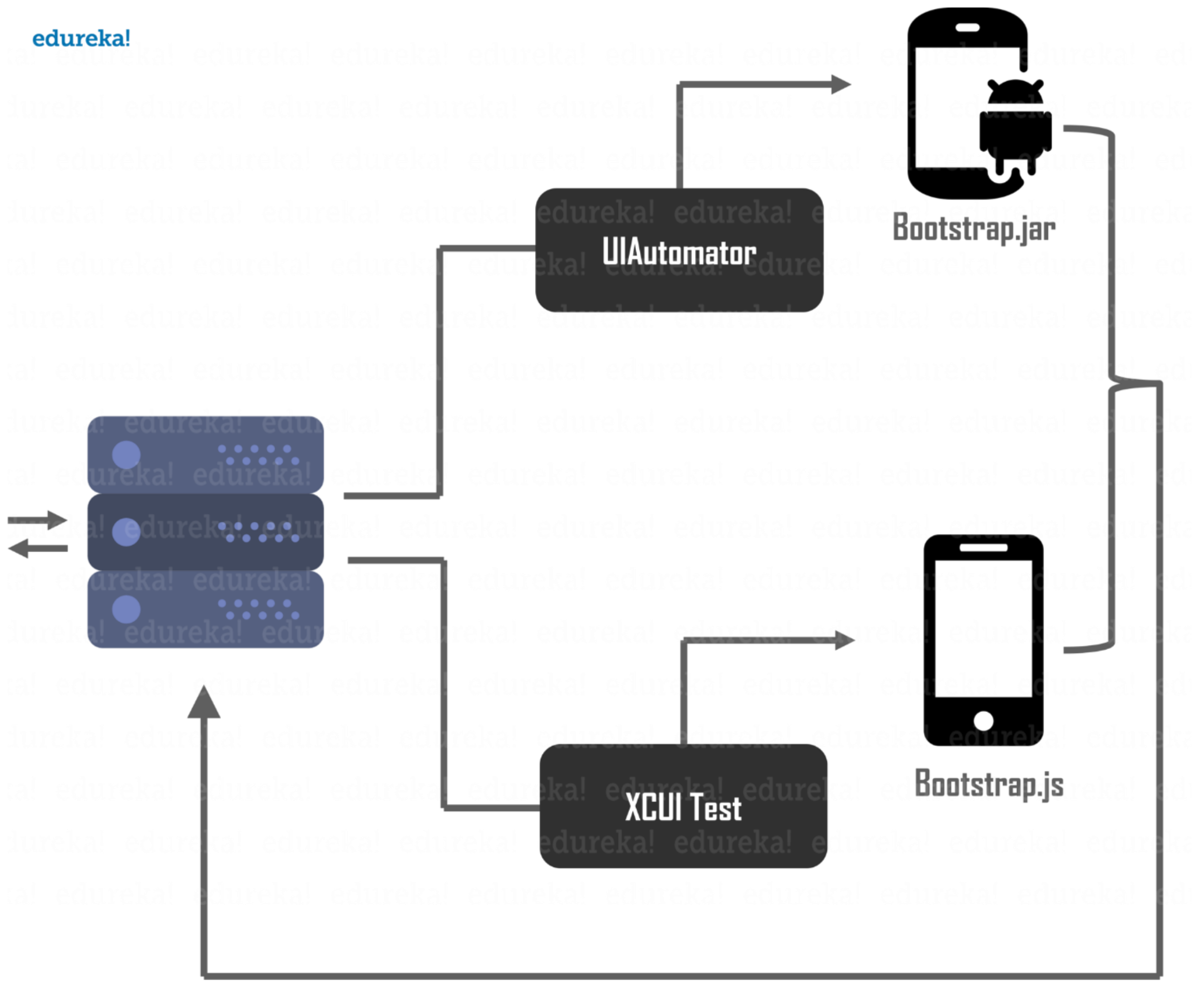
Hybrid applications merge the best of native and web applications. While these applications can be downloaded from the play store and can also access all the features of the device; they are in actuality a web application on the inside. They are developed using simplistic web-development languages like HTML and CSS. This allows them to run on any platform.

Below is a list of advantages of hybrid applications –

- They only have one codebase to manage
- App development doesn't take too long
- Easy to scale to another platform
- Access to complete device features just like a native app

Some example of hybrid apps are OLA, Basecamp, Instagram, Yelp etc.

# 8.Appium Architecture



- Appium is an *HTTP server* written using *node.js*
- The client communicates to the server using a session, where key elements of the communication process is sent with the help *JSON* objects. Communication is handled by the mobile JSON Wire Protocol.
- The server differentiates between an iOS request and an Android request using the *desiredCapabilites* arguments.
- *Appium* uses the **UIAutomator** test framework to execute commands on Android devices and emulators.
- *Appium* uses the **XCUITest** test framework to execute commands on Apple mobile devices and simulators.

- *Appium* uses **WinAppDriver** to execute commands for Windows Desktop apps. It is bundled with Appium and does not need to be installed separately.

- Appium server then processes the request to the respective UI Automators as seen in the **Appium architecture diagram** below.

- The UI Automator then processes the request and executes the command on a *simulator*/*emulator*/*real device*.
- The results of the test session are then communicated to the server and then back to the client system in terms of logs, using the mobile JSON Wire Protocol.

# 9.How Appium Works

## How Appium Works:-

- When we install the Appium, a server is also installed with it on our machine that exposes the REST API.
- It receives command and connection requests from the client and executes that command on devices like iOS or Android.
- It replies with the HTTP responses.
- To execute requests, it uses a mobile test automation framework to run the user interface of the app.

## How Appium works on different platforms: -

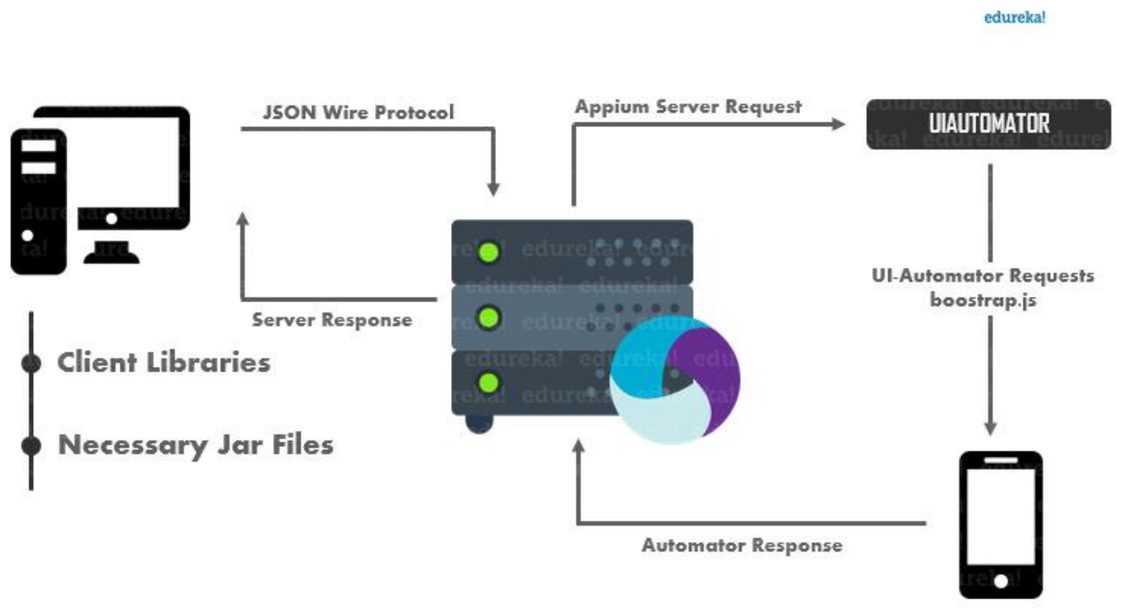
## a)Appium on Android:-

Appium on Android uses the UIAutomator framework for automation. UIAutomator is a framework built by android for automation purposes. So, let's take a look at the exact way that Appium works on Android.

1. Appium client (c/Java/Python/etc) connects with Appium Server and communicates via JSON Wire Protocol.
2. Appium Server then creates an automation session for the client and also checks the desired capabilities of the client. It then connects with the respective vendor-provided frameworks like UIAutomator.

3. UIAutomator will then communicate with bootstrap.jar which is running in simulator/emulator/real device for performing client operations.
4. Here, **bootstrap.jar** plays the role of a TCP server, which we can use to send the test command in order to perform the action on the Android device using UIAutomator.

The **Appium android architecture** diagram below gives a visual representation of the above steps.



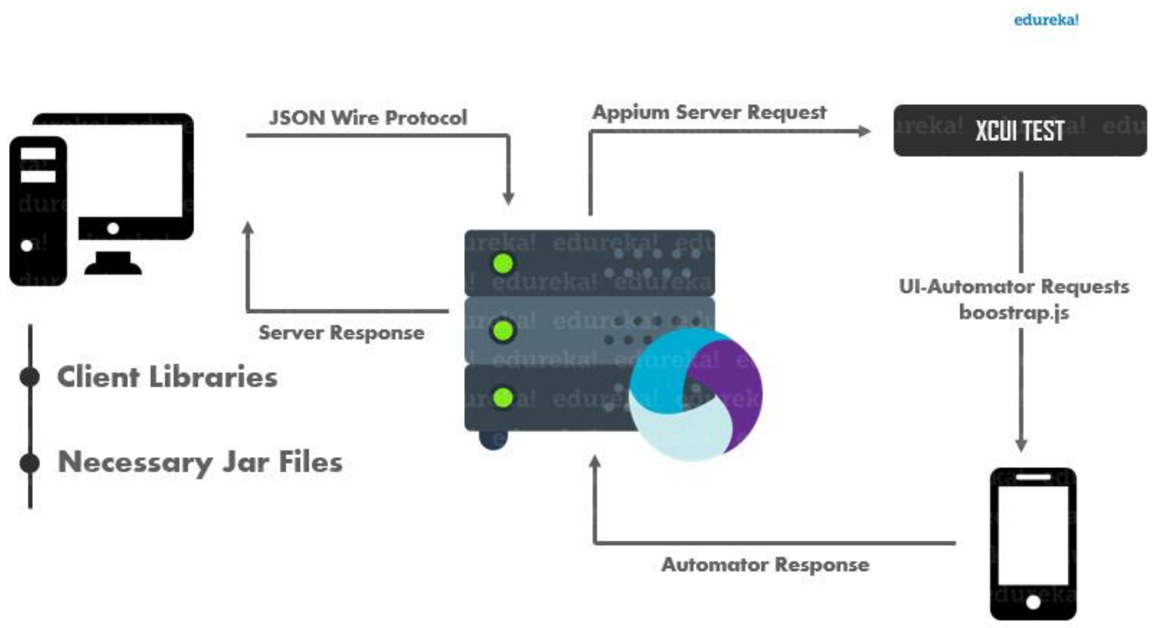
## b)Appium on iOS:-

On an iOS device, Appium uses Apple's XCUI Test API to interact with the UI elements. XCUITest is the automation framework that ships with Apple's XCode.

1. Appium client (c/Java/Python/etc) connects with Appium Server and communicates via JSON Wire Protocol.
2. Appium Server then creates an automation session for the client and also checks the desired capabilities of the client and connects with the respective vendor-provided framework like XCUI Test.
3. XCUI Test will then communicate with bootstrap.js which is running in a simulator/emulator/real device for performing client operations.

4. **Bootstrap.js** will perform the action on our application that is being tested. After the execution of the command, the client sends back the message to the Appium server with the log details of the executed command.

The **Appium iOS architecture** diagram below gives a visual representation of the above steps.



# 10.JSONWire Protocol

JSON Wire Protocol is a popular standard that facilitates communication between client libraries and the server in a heterogeneous system. Being a platform-agnostic and language-agnostic tool, Appium makes use of this feature of the JSON Wire Protocol.

JSON is a short form for JavaScript Object Notation, wherein complex data structures are represented. The wire protocol is a point-to-point data transmission mechanism.

JSON Wire Protocol, which is an extension to the Selenium JSON Wire Protocol.

In the Appium architecture, the communication between the Appium Server and client libraries is facilitated by the WebDriver that uses the JSON Wire protocol over an TTP REST request with JSON inputs. This arrangement uses the serialization and deserialization method to convert object data into JSON format, and vice versa.

JSONWire protocol contains an JSON Object which is created with key/value pairs.

## The JSON Protocol architecture:

1. Local End: It is the client-side of the protocol, normally client libraries written in a specific programming language.
2. Remote End: It is the server-side of the protocol
3. Intermediary end: Proxy systems that can act as remote end as well as the local end.
4. Endpoint node: It is the final remote end of the node structure.

The remote end is the server that reads requests from the local end clients and writes responses via a TCP socket.

# 11.Common Mistakes

- Not performing PoC on the application before the main automation phase.
- Not using best locator strategy that might increase the time taken to identify the element.
- In Android application, Appium won't identify the elements if it's not visible on page (those are present in page). So we need to swipe the page and check those elements.
- Missing to set environment variables for ANDROID_HOME and JAVA_HOME after installing Java and Android SDK
- Missing to map xcode path for the correct version of xcode as more than one xcode version is possible on the same machine.
- Skipping ideviceinstaller and iOS-deploy installation on Mac machine to execute scripts on iOS device. If there is below error even ideviceinstaller is installed properly on the machine, it can be resolved by setting the below value in project Run configuration Let me know if you need any further info Error: Could not initialize ideviceinstaller; make sure it is installed and works on your system(iOS) Name: PATH Value : /usr/local/bin:/usr/bin:/usr/sbin:/sbin.
- If robot class commands are present in the scripts and are executed on Mac machine, background java process would be started and a Java Cup icon is placed in the Dock, it causes the currently active window to lose focus. To resolve this, please specify the attribute "-Dapple.awt.UIElement=true" for jre.
- For iOS, only one device or simulator can be run on a mac machine at a time. Careful management and planning are required while scheduling tests across multiple devices at the same time.

# 12.Best practices to be followed for mobile app automation with Appium

## Proof of Concept:-

In order to have a successful mobile test automation process, it is highly recommended to have a pilot for all the applications as that would give an insight on the estimates and would uncover the challenges and help in proper planning. The following points should be kept in mind while implementing the POC:

- It should be implemented in critical scenarios to understand the scope as the time taken to automate is different for each app type and OS.
- It should be done on the elements that are unique to each OS. For example: the date and time picker.
- Dry run should also be performed to understand the efforts involved.

## Efforts Estimation:-

Efforts estimates should cover the following:

- Environment setup.
- Scripting the test cases with the challenges faced during the Proof of Concept.
- Effort required to make the script compatible with different Operating Systems and devices.
- Time taken for the “Dry Run”.
- Estimates should include time for analysing, development and unit testing of each scenario.

## UI Locator Strategy:-

A proper strategy should be in place to make Appium identify the element in the minimum time possible. The locator should also work if the hierarchy of the elements gets changed in the UI.

The order of the locators to be considered are as follows:

1. Id(Accessibility ID).
2. Name.
3. Value.
4. Classname.
5. Xpath.

# 13. GRAFT: Ours Ready Automation Framework for Testing

We own a self-crafted integrated accelerator which enables end-to-end automation of Android and iOS applications. It speeds up the automation of functional and regression testing by leveraging the existing automation tools thereby reducing the dependency on highly skilled testing personnel.

The accelerator brings down the initial development effort by 30% by utilizing the predefined methods in the framework. It brings in multiple other benefits including:

- Reduced overall cost due to the open source tools being used.
- Increased flexibility of time and resources. Reduced manual regression test effort by 60% to increase coverage of product areas.
- Increased software quality and reliability due to automated testing methods.
- Reduced defects and time-to-market. Reduced automated test development time and faster ROI realization on test automation.
- Improved test coverages by executing the scripts on cloud on different environments and platforms including OS versions.
- Reduced test automation development phase by over 30%.
- Reduced maintenance cost.
- Facilitates better communication between various stakeholders and developers, using tables for representing tests and reporting their results.
- Reduced dependency on technically skilled resources.

# 14.Framework

The different factors that decide the effectiveness of the framework are enlisted below:

- **Defined Folder Structure:-**

  Framework should have a well-defined folder structure to easily trace the items.

- **Portability:**

  The framework should work as expected with any folder hierarchy, any drive location and type of application.

- **Error Handling and Recovery Management:-**

  The process to be followed when an error or exception occur must be handled in framework rather than in the scripts.

- Functional libraries for general use and project specific purpose should be in separate folders for ease of maintenance.
- Test data to be maintained on a module basis to update them as and when required without touching the scripts. Also it is needed to ensure the security for the test data.
- The framework should be environment independent. By implementing few things that can be configured like browser, OS etc. we can make the framework environment independent.
- The selection of the test scripts to be executed and batch creation should be provided.
- There should be an option to select the environment details like OS and device versions along with the environments like emulator or simulator or device or cloud.
- There should be a mechanism to schedule the execution of batches.
- Test results should be easily understood by any tester with more details on the condition of the application under test and test data used. This will help the tester to reproduce the defect. Capturing the screenshot will also add value to decide if it is a defect or script failure.
- There should be a configuration mechanism to email the results to selected recipients.
- Scripting guidelines should be documented.

- Naming conventions and standard to be followed should be documented.
- Proper trace-ability should be visible to understand the coverage of the automation and execution of the defects.
- After the execution, the framework should provide some data for analysis and metrics preparation.
- Test framework should be easy to expand and maintain with fewer references to paid tools or other third-party libraries.

Above all these, there should be a continuous improvement plan to make the framework more matured. This will lead to a reduction in the manual effort by replacing it with the batch execution.

# 15.Competitors of Appium

There are several tools available for automated testing mobile applications, such as Robotium, Appium, Experitest, Selendroid, Kobiton, and Testdroid, etc. They all are tough contestants for Appium. But Selendroid and Robotium are one of the top competitors of Appium. Let us know some differences and see how they differ from each other.

## Appium vs Robotium:

- **Appium** is a cross-platform tool that supports both iOS and Android. Whereas **Robotium** only supports Android.
- **Appium** supports various languages while **Robotium** only supports Java programming language.
- **Appium** does not require application source code/library, whereas **Robotium** tool requires application source code or library.
- **Appium** can be used to test native, web, and hybrid mobile applications, whereas **Robotium** can only test native and hybrid applications.
- **Appium** supports many frameworks like Selenium. But **Robotium** is not compatible with Selenium at all.
- **In Appium**, you don't have to reinstall the application for a small change. But **Robotium** code leads to complete rebuild for a small change.

## Appium vs Selendroid:

- **Appium** is an open-source automation tool that supports both iOS and Android, while **Selendroid** is a test automation framework that only supports Android.
- In **Appium**, a small change does not require reinstallation of the application. But **Selendroid** requires reinstallation of the application.
- **Appium** has a strong and active community, whereas **Selendroid** does not have a strong community like Appium.
- **Appium** supports many frameworks and languages. On the other hand, **Selendroid** is compatible with Jenkin and Selenium.

- **Appium** does not require application source code/library, while **Selendroid** requires application source code or library.
- **Appium** supports all Android APIs with a limitation. Appium uses UIAutomator for tests running on API>=17, while for older APIs, it runs tests using Selendroid.
-

## Appium vs Selendroid vs Robotium:

| Appium | Selendroid | Robotium |
|---|---|---|
| Free and Open Source | Free and Open Source | Free and Open Source |
| Supports both iOS and Android | Supports only Android | Supports only Android |
| No need for app source code or library | Need app source code | Need app source code/library |
| Doesn't reinstall the application | Application reinstalled | Small changes in Robotium code leads to a complete rebuild |
| Supports multiple frameworks and programming languages | Compatible with Selenium and Jenkin | Supports java only. Not compatible with selenium at all |
| Strong & Active community | Community not as strong as Appium | Not such a proactive community |

# 16. System Requirements for Appium

## Minimum Hardware Requirements:

- Intel® i5 or i7 processor.
- 1 GB free hard disk space.
- 8 GB RAM
- 1 available USB 2.0 port and USB cable.

## Windows OS and Software Requirements:

- Microsoft® Windows XP™ SP2 (32 bit), Vista (32/64 bit), Windows 7 (32/64 bit), Windows 8 or higher.
- Net Framework 3.5 or higher.

## Mac OS and Software Requirements:

- OS X version 10.7 or higher.
- XCode and Command Line Developer Tools.

## Supported Devices:

- Android 4.4 and above.
- iOS 9 and above.

# 17.Pros & Cons of Appium Testing

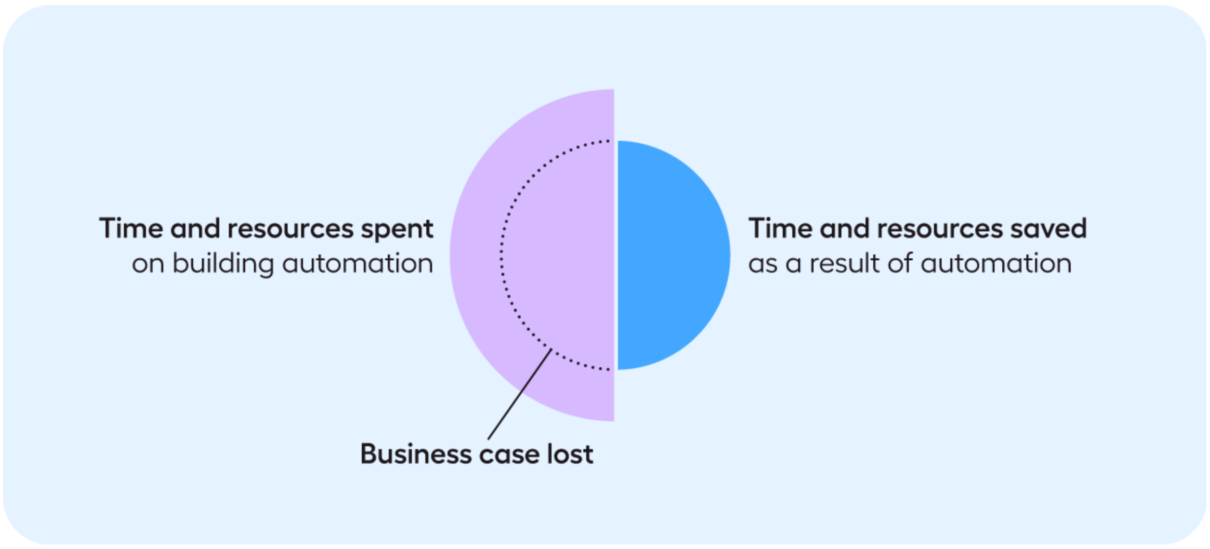
## The pros of Appium testing:-

- **Upfront costs**. Appium is a free, open-source framework. Setting up infrastructure in Appium won't cost much
- **Mobile testing**. It supports testing on native, web, and hybrid mobile applications
- **Operating systems**. You can test across iOS, Android and Windows apps
- **Programming languages**. It supports most programming languages
- **Device and operating system combinations**. You can test across hundreds of combinations of device types and operating systems (given you have the right plug-ins)
- **Set-up**. It's relatively easy to set up Appium

## The cons of Appium testing:

- **You have to code Appium tests**. A tester has to learn a programming language to be able to automate tests (which can take months). This prevents business users and testers from using automation. It also makes you reliant on developers or those with coding skills to build and maintain tests.

- **Heavy maintenance**. The scripts that you write in Appium have to be maintained. If you're using Appium for a small handful of tests, this won't require much time. However, if you're an enterprise that has to run mobile app testing as part of your regression suite, this could become a problem. If you don't regularly maintain the scripts, they will break. The maintenance alone can end up costing your business more than it should.


Time and resources spent on building automation
Time and resources saved as a result of automation
Business case lost

- **Naming conventions and gestures**. If your development team doesn't attribute elements with specific names and IDs, you'll find it challenging to code selectors to find the elements. If your development team hasn't been rigorous when adding IDs and specific names to elements, it can become very hard for selectors to work. Testing gestures in Appium is also challenging to code.
- **One-on-one support**. Compared to a test automation vendor, Appium does not offer one-on-one support. This can be challenging when you need to find a quick solution to a fault in the functionality of Appium. However, there is a community forum where you can seek answers to your questions.

# 18.Future Scope of Appium

## Why is Appium testing important in today's market?

Software testing is a very important and integral part of the mobile application development process. It helps the company to identify possible errors and bugs in the software and fix them completely before handing it over to the client. This improves the quality and user experience, allowing the company to enhance their reputation and brand and increase their sales to grow in the market. However, there are various reasons to start testing in Appium framework, which is as follows:

- Appium is completely free and open source framework which makes it very desirable and extensively used around the world.
- It is specifically designed and focus on development and testing of mobile applications, unlike any other framework.
- It is an Automation Testing tool that helps the user to perform testing on mobile applications without learning any developing skills.
- The framework can test native, hybrid and mobile web applications which can easily be tested on a mobile device or simulator.
- It supports several programming languages such as Java, C#, Ruby, Python, JavaScript, PHP, etc.
- Developing test scripts using Appium libraries are quite easy. It helps in developing or testing mobile applications for Android and iOS.
- Appium also supports various Internet browsers suggest Safari, Chrome and inbuilt Browser for Android.
- Appium generally works on these four philosophical points:
    - Should not recompile your application to automate.
    - Shouldn't be locked into a particular Framework or language.
    - It should be an open source.
    - And when it comes to automation APIs it should not reinvent the wheel.

During the past few years, the number of smartphone users has increased dramatically, and thus, companies are now targeting smartphone users to enhance their sales and revenue, for which, they require sales oriented and flexible mobile applications. By using **Appium testing application** development, agencies can provide business companies with the right tool that would help them better interact with their potential customers. Since most of the smartphone users belong in the category of Android and IOS, Appium Framework testing is considered the most groundbreaking and significant way of ensuring that the developed mobile application is worthy and meet all the requirements of the client.

All this tells the huge scope of **Appium testing** in the current business market and how it is growing on a daily basis. If you are willing to learn Appium testing and take advantage of this incredible field, then it is highly important for you to quickly join well-recognized training agency who would help you gain all the knowledge and skills regarding the testing framework. SLA Consultants India is well known to provide the **best software testing training course in Delhi**, Gurgaon, and Noida and targets college students who are either graduated in IT field or in their last year of college. The advance and industry oriented training course would also benefit working professionals who want to enhance their expertise further.

# 19.Conclusion

Indeed, by this seminar I came to know all the concepts of Appium, and I also understand that we use Appium in different ways for different types of Operating System (OS).

There are multiple options available today when it comes to automation. All the tools available would have their own advantages and disadvantages too. There are lots of benefits offered by Appium for automation testing. A single Appium test can run on multiple devices and OS versions. Creating Appium tests is also easy. But looking at the vast benefits that Appium offers, it is currently the best choice for automation testing.

I hereby also came to know that Manual Testing is so complicated & time consuming. It is very difficult to test the application against multiple users. So Appium is an automated testing tool to test the application in very short time.

# 20.References

https://www.edureka.co/blog/appium-architecture/

https://www.slideshare.net/EdurekaIN/appium-architecture-how-appium-works-edurekaa

https://www.javatpoint.com/appium

https://appium.io/docs/en/about-appium/intro/

https://www.edureka.co/blog/appium-tutorial/#:~:text=Appium%20is%20an%20open%20source,the%20mobile%20application%20testing%20domain.

https://www.slideshare.net/EdurekaIN/what-is-appium-edureka

https://www.geeksforgeeks.org/automation-tools-for-testing-android-applications/