Challenge 2

Given you are asked to propose an Automated Test Setup for **an iOS and Android Native App** Project.

- Please write a short(!) whitepaper in which you explain to the customer which tool you would recommend and why.
- Also, describe your ideal CI Pipeline to ensure the customer gets the best possible quality.

Automated Test Setup

Different mobile app testing frameworks offer different features and may confuse the app tester in choosing the best one according to the situation.
During the research, various open-source test frameworks were reviewed, that test mobile applications only on Android or only on iOS, and cross-platform test frameworks, that test mobile applications simultaneously on two platforms.
Below is written the chosen, most suitable test frameworks, their brief description, and pros and cons.

Test frameworks for iOS And Android

1. **Appium**

Appium is a leading open-source test framework that allows for cross-platform mobile native test automation, as it supports both iOS and Android platforms. It was derived from Selenium in an effort to extend automated testing functionality to mobile apps.

Pros:

- **Open-source:** It is open-source and hence free to use for everyone.
- **Cross-platform:** It's cross-platform in nature so you can use the same code among different operating systems. This also enhances the code reusability.
- **Programming Language Support:** Appium supports almost every programming language (like Ruby, Java, PHP, Node, Python) and every framework. So the app tester need not learn a new framework or language and can start testing right away.
- **No app recompiling:** Appium lives on the philosophy that the user should not be required to recompile the application or modify it for testing. Appium mobile app the testing framework facilitates the reusability of the code by using the same APIs for multiple platforms.
- **Tool independent:** It's flexible and can operate with a lot of tools.
- **Real device compatible**: Appium works with real devices, emulators, and simulators as well.
- **CI/CD integrations:** The framework can be easily integrated with continuous integration tools such as Jenkins.
- It **does not require** anything to be installed on the device

- Backed by a large and thriving community, Appium users get consistent support for troubleshooting their issues.

Cons:

- **Compatibility:** No backward compatibility for Android below 4.2
- **Speed:** Appium has been observed to be a bit slower than other testing frameworks in test execution.
- **Problems with locating elements:** Appium has difficulties locating elements and recognizing images. It just can't do that automatically. So, the team will have to enter the elements' positions manually.
- **Difficult setup:** Appium's setup process for Android testing is complicated

2. **Calabash**

Calabash is an open-source mobile app testing framework available for free to test mobile applications running on either Android or iOS. Calabash uses Ruby language to execute the tests, although it can easily do using Cucumber without any coding knowledge.

Pros:

- **Open-source:** It is open-source and hence free to use for everyone.
- **Cross-platform:** Support both Android and iOS
- **BDD-enabled:** Calabash uses behavior-driven development that is easier and faster to construct.
- **Programming Language Support:** Ruby
- **Real device compatible**: Calabash supports both real devices and emulators to execute automated test cases.
- **CI/CD integrations:** The framework can be easily integrated with continuous integration tools such as Jenkins.
- The framework is highly appreciated for being one of the most stable mobile app testing frameworks.

Cons:

- **Programming Language support limitation:** only Ruby supported. Ruby knowledge is required for writing automation test scenarios.
- Each platform has a separate tool
- Calabash server on Android can only be used to test the UI inside the application code. This problem does not persist with Appium
- Debugging the test scripts is often a major issue.
- The maintenance of the test data files would be difficult if more screens are tested by the scripts.

3. **Detox**

Detox is an open-source JavaScript-based test framework for React Native applications. It works cross-platform, accepts tests written in JavaScript, and works well with emulators and simulators. Detox is designed for gray box testing, allowing monitoring of the app from the inside, and is known to reduce test flakiness as it is well synchronized with the app's activity.

Pros:

- **Open-source:** It is open-source and hence free to use for everyone.
- **Cross-platform:** Support both Android and iOS

- **CI/CD integrations:** Detox can run end-to-end tests integrating with CI tools such as Travis very smoothly.
- **Real device compatible**: Detox tests the mobile application after running it on the simulator/emulator, which resembles a real user-like testing behavior.

Cons:

- **Programming Language support limitation:** only Javascript supported. Javascript knowledge is required for writing automation test scenarios.
- Detox is a JavaScript mobile testing framework that is built into the application in order to allow test execution to begin when the app is launched.

Test frameworks for Android

4. **Espresso**

Espresso is a Google-made Android testing framework and has been a popular choice due to its high performance. Espresso believes in creating very simple and straightforward tests without worrying about the application's infrastructure. Furthermore, it is open-source, which gives the developers the power to customize the framework. It is extremely fast and has the least test execution time and fallibility.

Pros:

- **Open-source:** It is open-source and hence free to use for everyone.
- **Stability and speed:** Espresso is comparatively stable and faster in test execution.
- Espresso allows compiling automated Android UI tests into a separate APK. This means that the test suite will run next to the app on the device, which is very convenient. On top of that, because Espresso doesn't require server communication, it provides feedback faster than Appium.
- **Compact API:** Espresso has a simple and lightweight API with three components: viewMatchers, viewActions, and viewAssertations.
- **Setup**: The setup process for Espresso is much more straightforward as compared to Appium

Cons:

- **Programming Language Support:** Espresso only supports Java and JUnit programming languages, which makes it a perfect tool for Android developers who are used to creating native apps. At the same time, it's a drawback since it means QA engineers are limited in their stack.
- **Cross-platform:** Only Android platform supported

Test Frameworks for iOS

5. **XCUITest**

XCUITest is considered the primary and most popular iOS test automation framework for testing an iOS mobile application. The iOS testing framework uses instance methods and creates a friendly environment for the iOS app developers. XCUITest uses Objective-C and Swift programming languages for testing and is compatible with XCode 5.0+.

Pros:

- **Open-source:** It is open-source and hence free to use for everyone.

- **Stability and speed:** The XCUITest framework was specifically designed with the aim of simplifying UI testing for iOS apps. As the framework is dedicated to just UI testing of iOS apps, it ensures that the tests are performed exceptionally well. Thus tests are stable, highly reliable, and performed faster.
- **CI/CD integrations:** XCUITest gives good control over continuous integration facilities.
- **Setup**: The setup for XCUITest is more straightforward as compared to Appium

Cons:

- **Programming Language Support:** XCUITest cases can only be written in either Swift or Objective-C programming language.
- **Cross-platform:** Only iOS platform supported

Conclusion:

As a more preferable solution, it is recommended to use the **Appium test framework** for solving the current goal that covered almost completely all requirements:

- Appium flexibility and a variety of supported programming languages allow faster get familiar with the test framework or find QA engineers who are experienced in this area.
- Also, Appium helps to save time for developing new and supporting already existing tests for both platforms.
- Ease integration with continuous integration tools allows maintaining high product quality.
- Appium makes a guarantee of the continued growth of test framework and problem-solving for future test support.

CI/CD Pipeline

CI/CD Pipeline should include the following items:

- Maintain different images for Android and iOS platforms.
    - At first, it should include the latest version of Android and iOS, and the most used.
    - Then, expand the library of images with more specific images or older ones.
- Support the possibility to perform tests on real devices.
    - At first, include support more popular models of devices.
    - Then, add the possibility to communicate with more specific device models.
- Maintain flexibility:
    - allow choosing needed images with default or specific parameters, for example, CPU, Storage, available disk space.
    - allow choosing needed tests to the test run. For example, perform tests for new functionality and specific tests from regression scope.
- Environment preparation module that makes the environment test ready. This stage includes some specific steps that needed to do before the test run, for example, device/image configuration, installing a mobile app with the needed version, copy tests and etc.

- Collecting test run results in reports for analyzation test run during acceptance testing new functionality or nightly test run.

Continuous integration server can be:

- Open-source, for example, **Jenkins** - the leader of CI servers, is free to use.
- Partially free or completely paid. For example, CircleCI, Travis CI, Bitrise and etc.