AUTOMATED TEST SETUP FOR IOS AND ANDROID NATIVE APP

SUGANYA KANNAN

CONTENTS

1 INTRODUCTION

Given the wide range of devices, running on different versions of the operating system, the application(app) testers have to test their application in all versions available to maintain reliability of the app. This consumes a lot of time and effort. This leads to higher porting costs and lower profits since the developers need a longer time to market the application.

The right automation tool setup depends on several criteria such as the functionality of the application. A complete analysis of the application regarding what needs to be tested and what scenarios of these testing could be automated should be done. This analysis will help us in choosing the right tool. Since software testing is a team effort, we also have to consider which tool our team members are comfortable with or they enjoy learning such that the learning curve is not very steep. It would be a good idea to have Test APIs provided by app stores to help testers check conformity to the guidelines. This would help avoid the waiting period that comes with submitting the app to the market for the review process. However, care has to be taken to ensure that these future test APIs are foolproof and without loopholes.

In this paper, I try to analyze a few testing tools with their pros and cons. I have also written my idea bout the Continuous Integration (CI) pipeline which ensures the best quality for the application

2 TOOLS CONSIDERED

I have taken these three tools based on their popularity and their widely available community support. Automation tools can be compared based on several features such as their support for both Android and iOS apps, support for both native and web mobile apps, logging, integration with cloud and other testing tools, the possibility of parallel execution of testing scenarios. Their flexibility to support several programming languages and also the support to set up a CI pipeline adds great value to them. The ability to create resourceful test reports is also an important feature.

2.1 Appium

Appium is a popular open-source test automation framework that supports testing in Android and iOS applications. It has wide community support. It is easy to learn as it is similar to the selenium automation framework. It supports several programming languages. It also provides parallel execution of tests and supports continuous Integration. One of the disadvantages is that it does not support older Android versions. Keeping that aside and the initial setup needed for configuration it's one of the best available automation frameworks.

2.2 Espresso

Espresso is an automation framework solely for testing Android applications. It is easy to set up and very reliable. Element identification is made easy with Espresso. The main disadvantage is that it supports only Java. If we have to test only Android applications, this is one of the best possible options.

2.3 XCUITest

XCUITest is an automation framework to test native iOS applications. It is supported with Appium and provides several functionalities in addition to those provided Appium libraries. It mainly focuses on UI testing and can also identify the scenarios which were overlooked to be tested using Appium. It helps to make our applications accessible. One disadvantage is that testers who don't know SWIFT will have to spend time before starting to use this framework.

3 continuous integration(ci) pipeline

In an agile environment, the requirement keeps changing. It should be made sure that the software is verified and validated every time an iteration happens. In the case of bug fixes, regression testing is important. To make this testing efficient, the CI pipeline should be employed in a way that the tests that can be executed without human intervention could be triggered right after a feature commit or a bug fix. The quality attributes of the code can also be checked using several static code analysis tools integrated with the CI pipeline. The CI pipeline should be made secure and every time the pipeline is getting executed it should not bear the residual effects from the previous executions. Using a container environment such as Docker will help us in achieving this. It should be possible to roll back to previous versions. Several automation testing tools provide support for continuous integration For example we can use the Jenkins tool together with the Appium framework to create a CI pipeline. The pipeline can also be configured to run on different servers and also on the cloud environment. Several information about the builds can also be acquired in the form of reports. Emails can be triggered to the respective team in-case of build failures.