

Overview:

The client offers both Android and iOS applications through which people can get auscultation consultations online from experts by recording and sharing recorded sounds. End users can also listen and get trained to recognize different pathologic murmurs and get treatment. The client was facing issues with maintaining the quality of applications. With progressive releases, a lot of defects were making it to the live applications affecting the quality and ratings of the application. The client approached Mindfire solutions to build an automation framework running daily builds over CI/CD sharing daily status reports of health for both the applications.

Client details

Client Name: Confidential | **Industry:** Healthcare | **Location:** Canada and Austria

Technologies

Appium (Mobile Automation), Selenium WebDriver (UI Automation), Cucumber, Extent Reports, Java, AppleScript, Eclipse, Maven, Platforms - Server - MAC OS X , Devices - Android Emulator, iOS Simulator, Real devices

Project Description:

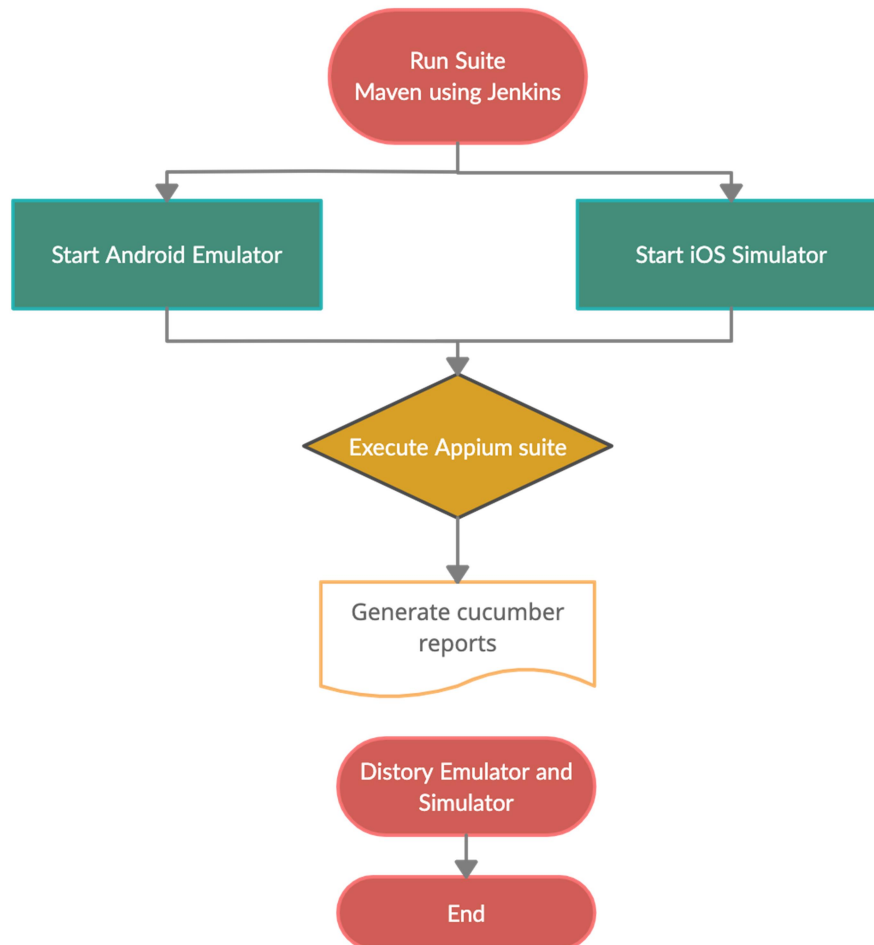
The client is a leader in AI and auscultation detection technologies using mobile applications. To keep up with mHealth industry, the new requirements were pushed by the development team after every sprint. The QA team was testing the application but given the nature of fast-paced changes, was not able to always completely regress the applications over multiple Android OS and iOS versions. The defects in applications were causing an impact on the ratings of the applications. The client approached Mindfire Solutions with the requirement to build an automation suite which can perform regressions on applications on a daily basis. The intent was to get a report of the health of the release, give insights to product to be in a position to decide whether to proceed with the release or not.

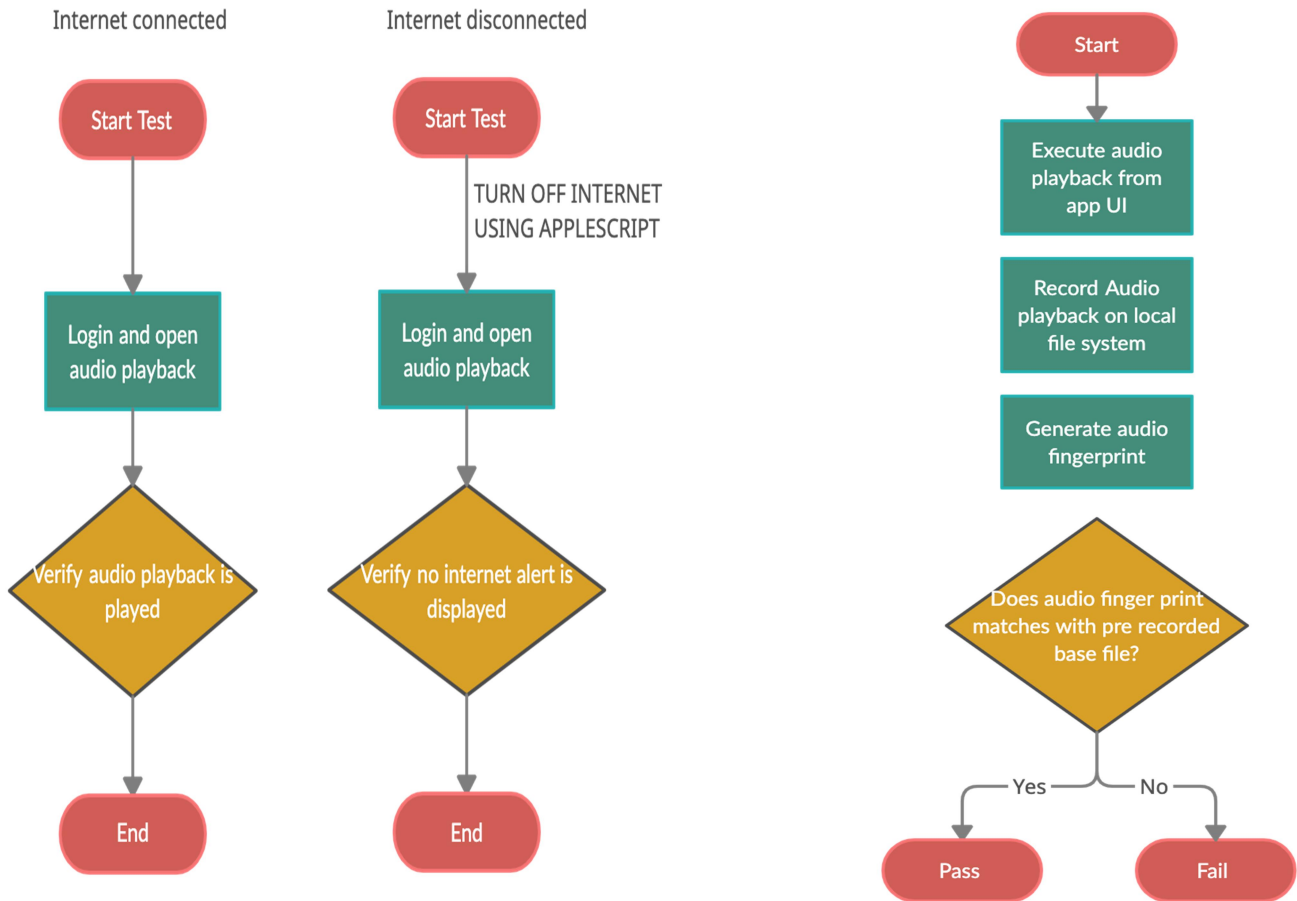
Our team started discussions with the client to get a better understanding of the issues with the existing QA process. The client wanted a complete end-to-end automation of the application. The solution needed to be robust and easy-to-maintain. It had to cover all of the scenarios that were getting executed by the manual QA team. Some of the silent features of our solutions are mentioned below with elaboration:

- The solution needs to verify not just when the end-user is connected to the internet but also with bad or no internet at all. The application would behave unexpectedly when a user runs out of internet connectivity.
- Automation should be able to verify the audio playback throughout the application. Since all applications were dependent on making sure the end-user can get training by listening to audio output, this was critical functionality for the client. For this we recorded the base audio and also the playback during runtime, and then audio fingerprinted both sounds to get a match.
- The suite should be able to verify both iOS and Android applications with the same code base. We provided a page factory model so the same elements can be targeted from code depending upon the platform targeted. The suite should be able to run multiple iOS and Android OS versions.
- Automation should be able to handle the dynamic WebView used in an application and verify the HTML is displayed as expected.
- The client should be able to execute all or only a small subset of test cases. This was made possible using cucumber tags.
- The solution should be data-driven where inputs can be updated when required. We used cucumber with data tables and scenario outlines for test data.

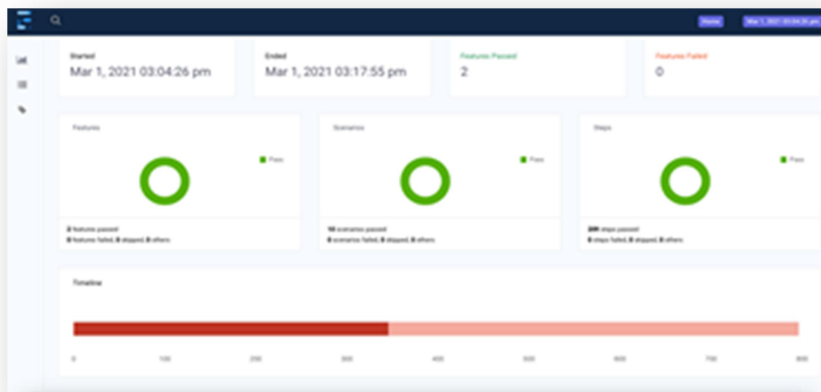
- The emulator or simulators should be created by the automation and once the suite has completed, the same should be cleaned up.
- The suite should be able to execute on CI/CD for daily builds. We used apache maven and provided integration with Jenkins where clients can execute builds and see results.
- The report generated should not be technical, should be in plain English language (we used gherkin language), should contain images of all interactions automation performed with the application since the consumers would both developers and business owners.

Architecture Design:

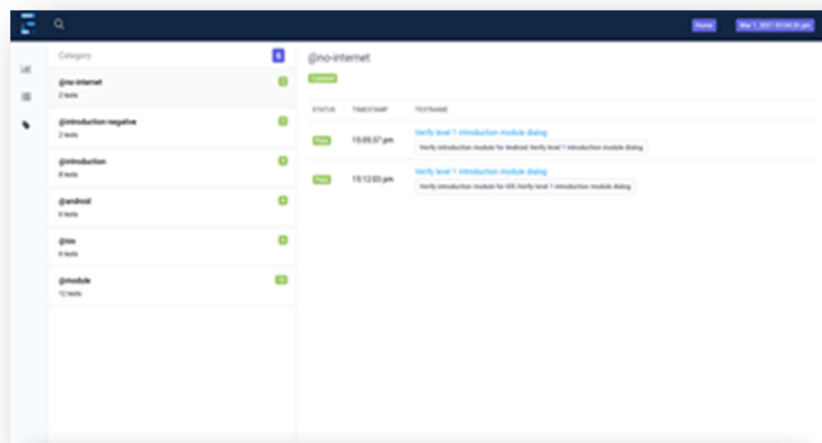




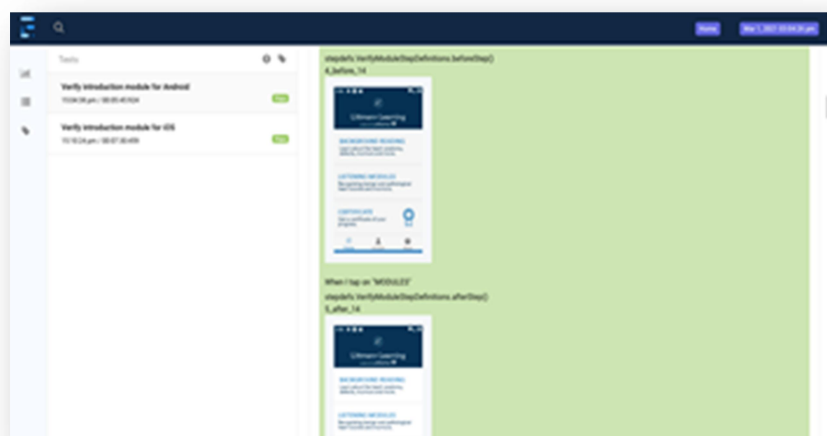
Screen shots:



Screen 1: Consolidated report



Screen 2: Tag wise results



Screen 3: Detailed results with mobile application screenshot