mindfiresolutions
BURN IGNORANCE

Case Study

Automated testing solution for IoT based project

Executive Summary

Our clients, a California, USA based smart and innovative IT solutions provider (Healthcare Management industry) wanted automated testing solutions for all the applications they have , which take major roles on delivering the control and usage of the smart IoT product to the end users. An IoT device that often interacts with cloud servers to record statistics and to get the various operation commands from user is backed by REST APIs, which is the major building block of all the customer facing applications on different platforms e.g Web, Android, iOS etc. Automating Http based REST APIs which needs the complete understanding of functionality/ business logic of the project itself was a challenging and well experiencing task for us. We did automated UI and functionality testing of all the applications built for Android, iOS and Web.We provided an automation solution which makes sure that every new build has minimal defects by providing reports for Bugs. We provided solutions like test case updation and maintenance of the script for the new builds of the application with added features at a later stage. Automated test of APIs (consumed in all the applications which are customer facing), opened the door for very early detection of defects - even before we find defect on UI level. Automated UI testing made sure that customer/end user gets a very easy to use and robust interface with minimal glitch.

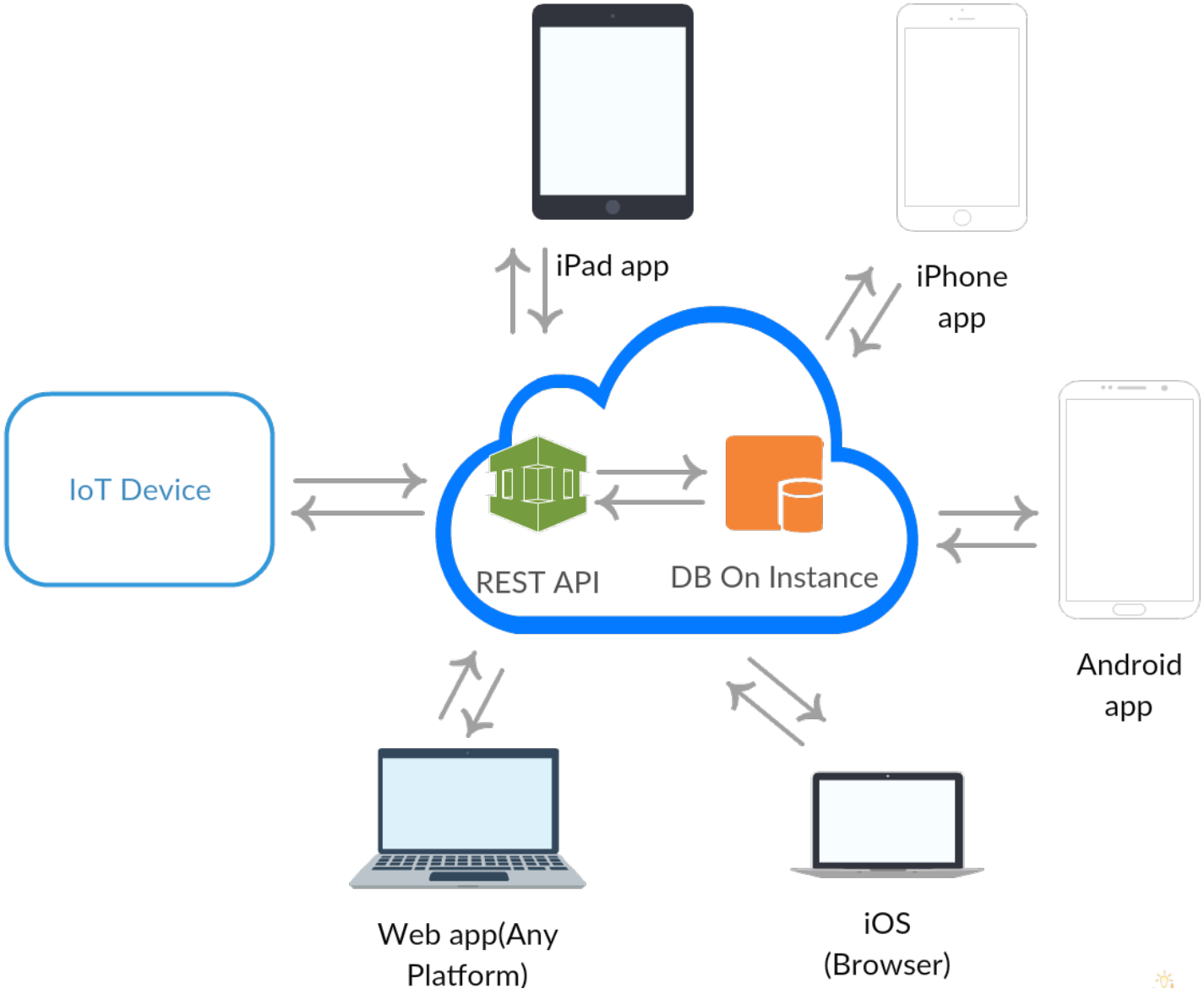
About Our Client

Client: A major innovative and smart IT service provider in Healthcare management domain (IOT based) | **Location:** California, USA | **Industry:** Healthcare Management

Business Situation

Our client wanted automated testing solutions for the applications , which are the integral parts of the IoT product they have. When a consumer uses the IoT device , all the functionalities , data statistics and control of the device resides with the cloud server to which the IoT device is connected to. There are applications on different platforms which provide user friendly UIs to consumers to get the record of usage and to control the device right away from the application itself. Besides this there was an application which is used for the service person to diagnose the components of the IoT device and configure those to make it ready for the user. REST APIs they have , which has a major role on delivering various crucial data to the customer facing applications made for different platforms. They wanted the project to be hassle free, which can be easily executed by anybody from their end and to output easily interpreted test results. They looked for fully automated solutions that would be a suitable candidate for integration with

Continuous Integration tools like Jenkins. Our client wanted these regression suites to be flexible enough to be executed with any of the testing environments along with production itself, with any set of credentials. Not only the validations were to be done from a set of expected results , out client wanted data to be matched from the application database on real time basis.



Our Solution and Customer Benefits

We have a number of applications based on a number of platforms to be automated. At the same time we had applications which have different purposes of use. Moreover for each platform we had many automated testing tools/libraries available.

After analysing and understanding the core workflow of the project , we narrowed down to the following solutions.

Web Application: Selenium WebDriver

Android: Robotium

iOS: Appium

REST API: Apache HttpClient with supporting JSON parsers

We suggested an approach of integrating above Java based automation libraries with a powerful framework which is built with the following features.

Brief description of these individual tools/libraries and the purpose of use in the framework:

1. One of the following tools , according to the application under automation:

Selenium WebDriver
This Java library contains the methods/APIs to develop test scripts which can interact with the browser and based on the success or failure of any operation a specific test method is marked as pass or fail.

Appium Java-Client
This Java library contains the methods/APIs to develop test scripts which can interact with the browser/app and based on the success or failure of any operation a specific test method is marked as pass or fail. Basically Appium Java Client is the implementation of WebDriver API.

Robotium
This Java library contains the methods/APIs to develop test scripts which can interact with the mobile application and based on the success or failure of any operation a specific test method is marked as pass or fail.

Apache HttpClient
This library has methods to do various Http operations and call methods like get, post etc and extract the response details. So this library is responsible for all the request and response stuffs with the APIs we test.

2. Maven
Maven is used for two purposes - Project Building and Dependency Management
All the Java libraries used across the project are mentioned in pom.xml file.As we are building the project from Jenkins , Maven helps building the project smoothly in Jenkins.

3. JUnit
We have used JUnit to define the order of execution of test methods. Basically JUnit is implemented in DriverScript.java. For reporting and controlling the execution of test methods we have the customised hybrid framework.

4. Log4J
Log4J is used for logging. Each operation that is done using HttpClient is logged to both console and a log file 'application.log'. Log4J configurations are set in log4j.properties file

5. JXL
This library is used to read excel files/spreadsheets. Basically we have two xls files under Config package. controller.xls contains TestSuites and Test Methods. Here we can select which test suite or which test method to run and which methods to skip. testData.xls contains various data which is used across the automation to input details to various fields etc in the application or to assert/ verify various objects/Strings while testing.

6. Javamail
This is used to email reports after the test run.

7. JDBC Drivers
These drivers are used to interact with DB. Basically we use FileMaker JDBCDriver to import test result to the QA Test Case Application. OJDBC driver is used to connect with DB to fetch details on accounts and other details to be asserted with the API response or to be attached with the API request.

8. org.json
This library helps to parsing information of the HTTP request and response components. As almost all the APIs we test have JSOn request body and response we get JSON as well, hence this library is used to build the request JSON object and response is too parsed to a JSON object in order to verify/assert different components , Key value pairs of response JSON.

This approach best suited the requirement of CI support and better test coverage.

We wrote test cases covering positive ,along with negative test scenarios which helped protecting APIs from a security point of view.

We validated test expectations from the application database using JDBC.

We prepared documentation containing instructions to handle script running in Mac OS and configuration details etc.

Divided the test suites according to the functionality of the application to best suite Job configuration with Jenkins. We configured the Jobs with a VM provided by client using Share Screen and configured that machine to be friendly with the script run.

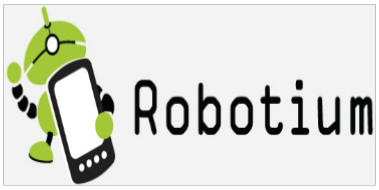
Our automation test reports gets generated in HTML, Spreadsheet and csv format as well. Demonstrating test results test case wise (along with complete request and response details), which helped client to figure out the exact portions of the application which has defects.

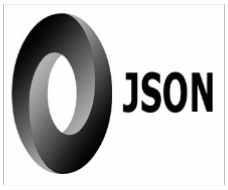
Later we integrated our automation test suite with a test case application which made defect tracking process an automated solution. Whenever a test case fails , it saves a record in test

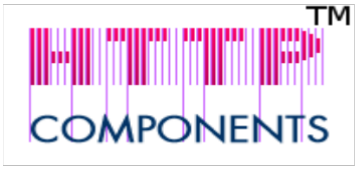
case application database and at any point of time , we can track which build had the issue on which date.

Technologies and Tools

Selenium WebDriver, Robotium, Appium,Apache HttpClient,org.json(JSON Parser), Java, Eclipse, Maven, Jenkins

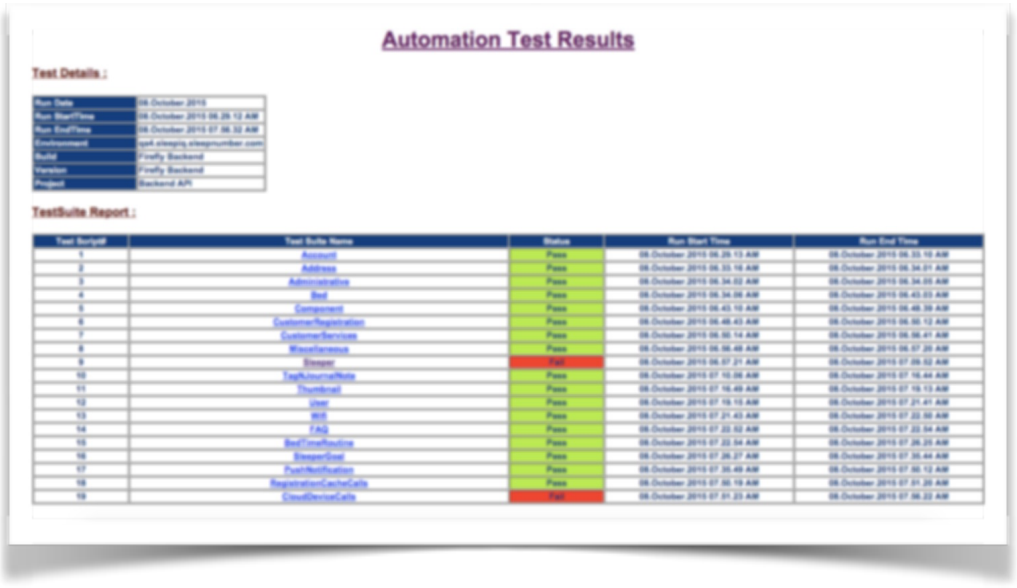






Final Result

Glimpse of project being developed using Eclipse IDE



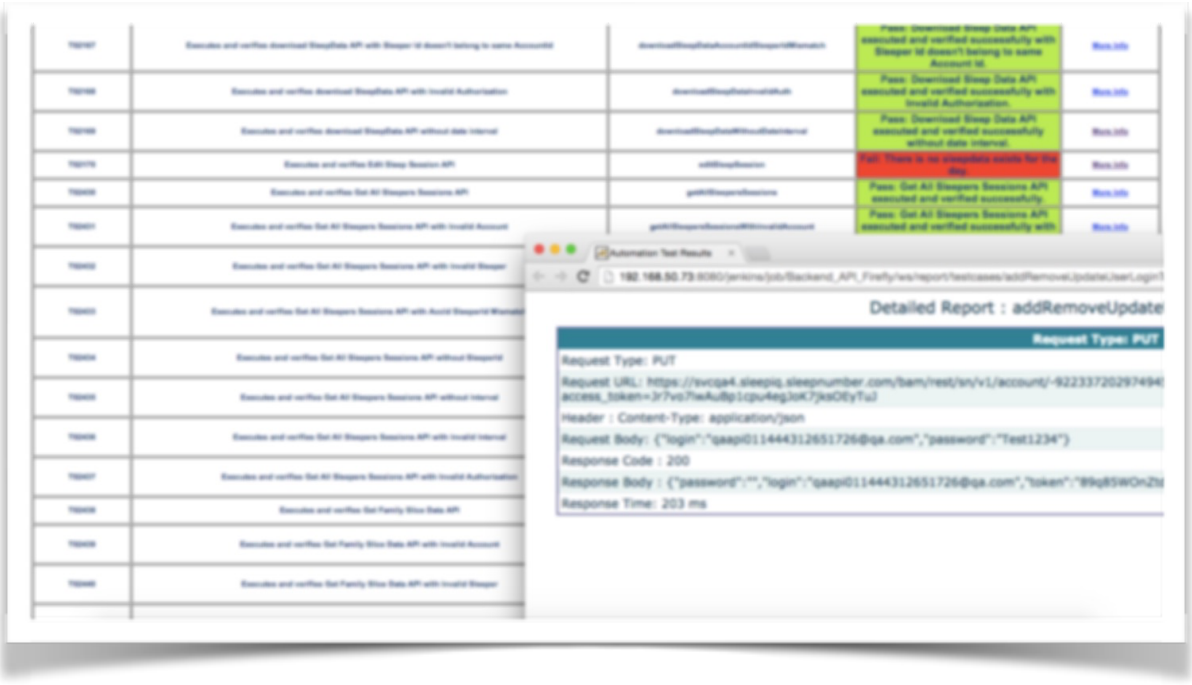
Automation Test Results

Test Details :

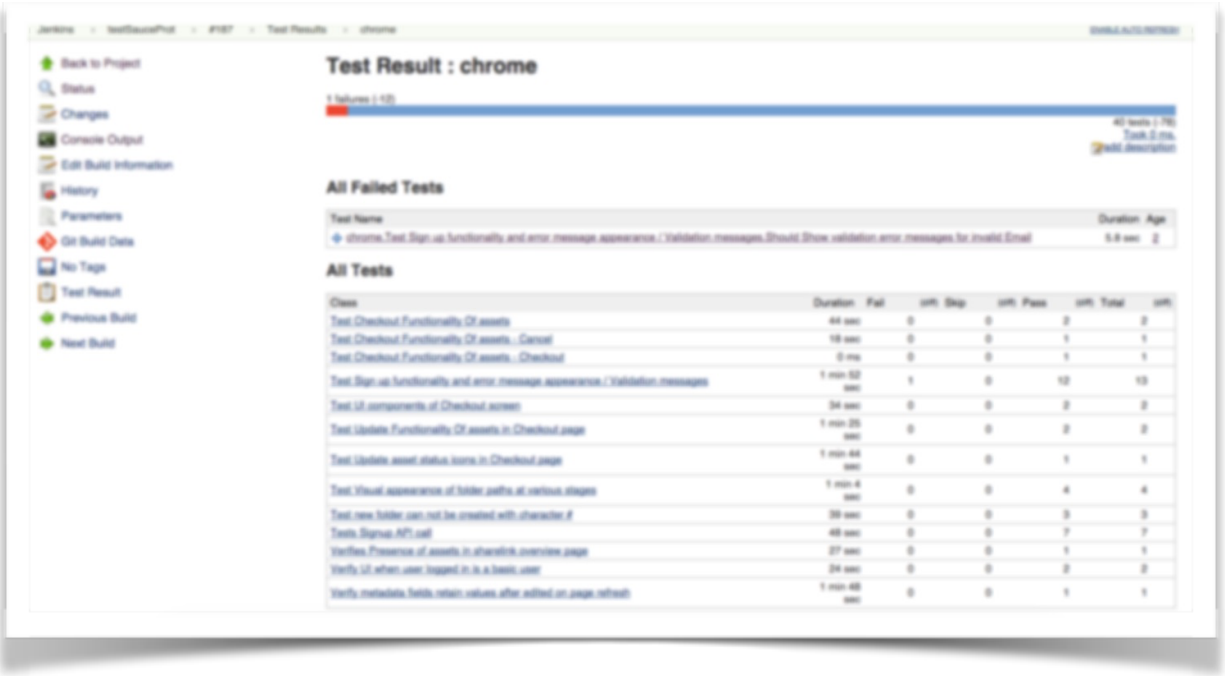
Run Date	06 October 2015
Run StartTime	06 October 2015 06:28:12 AM
Run EndTime	06 October 2015 07:06:32 AM
Environment	qa4.sleepiq.sleepnumber.com
Build	Firefly Backend
Version	Firefly Backend
Project	Backend API

TestSuite Report :

Test ScriptID	Test Suite Name	Status	Run Start Time	Run End Time
1	Account	Pass	06 October 2015 06:28:13 AM	06 October 2015 06:33:10 AM
2	Address	Pass	06 October 2015 06:33:10 AM	06 October 2015 06:34:01 AM
3	Administrative	Pass	06 October 2015 06:34:02 AM	06 October 2015 06:34:35 AM
4	Bed	Pass	06 October 2015 06:34:36 AM	06 October 2015 06:43:03 AM
5	Component	Pass	06 October 2015 06:43:10 AM	06 October 2015 06:46:39 AM
6	CustomerRegistration	Pass	06 October 2015 06:46:43 AM	06 October 2015 06:50:12 AM
7	CustomerServices	Pass	06 October 2015 06:50:14 AM	06 October 2015 06:56:41 AM
8	Miscellaneous	Pass	06 October 2015 06:56:48 AM	06 October 2015 06:57:20 AM
9	Sleeper	Fail	06 October 2015 06:57:21 AM	06 October 2015 07:06:52 AM
10	TagAJournalNote	Pass	06 October 2015 07:10:08 AM	06 October 2015 07:16:44 AM
11	Thumbpad	Pass	06 October 2015 07:16:49 AM	06 October 2015 07:19:13 AM
12	User	Pass	06 October 2015 07:19:15 AM	06 October 2015 07:21:47 AM
13	Will	Pass	06 October 2015 07:21:43 AM	06 October 2015 07:22:50 AM
14	FAQ	Pass	06 October 2015 07:22:52 AM	06 October 2015 07:22:54 AM
15	BedTimeRoutine	Pass	06 October 2015 07:22:54 AM	06 October 2015 07:26:25 AM
16	SleeperGoal	Pass	06 October 2015 07:26:27 AM	06 October 2015 07:35:44 AM
17	PushNotification	Pass	06 October 2015 07:35:49 AM	06 October 2015 07:36:12 AM
18	RegistrationCacheCalls	Pass	06 October 2015 07:36:19 AM	06 October 2015 07:51:20 AM
19	CloudDeviceCalls	Fail	06 October 2015 07:51:23 AM	06 October 2015 07:56:22 AM



HTML Reports



Report on Jenkins

Application Snapshots

Customer Benefits

The powerful and easy-to-use automation script helped client in the same fashion they wanted it to be. Smooth Job execution from Jenkins and finding out failures in a detailed manner in reports facilitated to figure out the problems in more easier way, avoiding the exhaustive manual effort.

Testing hundreds of APIs and covering both positive and negative scenarios manually consumes a lot of time. Automation testing of APIs makes the regression testing superfast with best possible test coverage. So this helped our client to find defects in just 15 minutes of script run whenever a new build comes in.

Future Relationship

A work of new of its kind and successful execution of the project led us for a strong business relationship with a couple of other projects. Along with this we got the chance of maintenance as well as enhancement of script according to the features added in the further builds.