# Test Automation for Next-Generation Interfaces

**Authors: Andrew Morgan, Sanil Pillai, and Anu Raturi**
andrew.morgan@infostretch.com**,** sanil.pillai@infostretch.com**,**
anu.raturi@infostretch.com

## Abstract

Today's IT systems communicate with customers through multiple points of engagement and various interfaces, ranging from web, mobile and voice, to bots and apps like Alexa or Siri. These systems need to provide seamless hand-offs between different points of interactions, while at the same time providing relevant and contextual information at speed. To accomplish this, your project team must be able to successfully pair device hardware capabilities and intelligent software technologies such as location intelligence, biometric sensing, Bluetooth, etc. Testing these systems and interfaces is becoming an increasingly complex task — traditional testing and automation processes simply don't apply to next-generation interfaces.

This white paper will shed light on the landscape of these new, hyper-connected systems and their testing nuances, help you understand next-generation testing challenges that you need to address, and share the best strategies on how to effectively test them.

## Biography

*Andrew Morgan is the Director of Product Marketing at Infostretch. He is an experienced leader in strategic analysis, opportunity assessment, and roadmap execution. With his guidance and expertise, he has helped companies expand their digital initiatives to unprecedented levels.*

*Sanil Pillai is the Head of Innovation at Infostretch. He is an experienced engineering leader for digital and enterprise applications. He has built and managed offshore and onsite engineering teams, managed mobile projects for Fortune 500 clients, and has deep technical and functional expertise. At Infostretch, Sanil has established agile development and Continuous Integration methodologies, tracking metrics and monitoring processes to ensure continuous improvement in the development organization.*

*Anu Raturi is the Director of Infostretch Labs. He is an experienced engineering leader for digital and new tech applications. He has built and managed offshore and onsite engineering teams for Large Scale Companies as well as startups. At Infostretch, Anu has established innovative technologies in Automation and AI practices to ensure continuous improvement in the development organization adapting technical innovation.*

# 1 Introduction

In today's hyperconnected world, connectivity is not limited to the mobile device alone. A plethora of smart devices and variety of sensors have expanded the connected ecosystem. Applications are increasingly leveraging data collected from peripherals like camera, BLE (Bluetooth Low Energy), etc. to add new capabilities. Testing such new age applications is challenging as current testing frameworks are not capable of testing the end-to-end functionality of such applications. Today, the enterprises are no longer confined to mobile and web presence alone to connect with customers. Consumers have shifted to smart devices and wearables and expect the companies to serve them on all the channels. Hence, it has become mandatory for enterprises to have an omnichannel presence so that they can reach out to maximum number of consumers. Rising consumer expectations and new business models like Phygital – 'Seamless integration of online and physical channels' have led enterprises to develop applications across a range of new interfaces such as wearable devices, chatbots, voice bots, virtual reality, smart TV applications, etc.

Today, applications have become the face of the organization to interact with their customers. So while developing these applications, it is equally important to ensure that these applications work perfectly. Any service failure due to undetected defects during the development and testing phase is a huge reputation risk and can harm the brand image of the organization. In this white paper, we look at the various challenges encountered by enterprises to ensure that the new age applications are able to function correctly. This white paper discusses approaches to overcome the testing challenges posed by the addition of inputs from peripherals like camera, Touch ID, Apple Pay, GPS, BLE, etc.

This white paper also defines the approach to test chatbots and voice bots which have gained tremendous adoption in the recent years. Chatbots are now becoming common and are now available on all major platforms such as Skype, Facebook, Slack, etc. Similarly, adoption of voice assistants like Amazon Echo and Google Home has also increased multifold. This has compelled the enterprises to reach out to their customers through these interfaces. This white paper discusses the challenges faced while testing these bots as well as defines the major factors to be considered while testing them.

# 2 Mobile Automation

Agile practices and DevOps have put Test Automation into a very different ball game from an automation coverage perspective. Higher coverage will definitely help faster release of better, quality products.

Mobile technology has made giant leaps with hardware capabilities including faster CPUs, larger memory and built-in sensors. With customers going digital, technology change has created new business models as well as new use cases around leveraging new hardware capabilities. Almost all mobile applications have at least one feature using device hardware capabilities. Popular frameworks like Appium have limitations in automating test scenarios involving mobile hardware, sensors etc.

A couple of scenarios involving device hardware like camera, Bluetooth, touch sensor is mentioned below:

- Check deposit features using mobile banking application enables the user to deposit check by capturing check image using camera. Data like the amount, payee, etc. is captured using OCR (Optical Character Recognition). Testing of this feature includes scenarios like scanning a check with a different amount, different handwriting or maybe a blank check. Automated end-to-end scenarios coupled with different test data can validate the functioning of the application across multiple devices and operating systems combinations. There is also a need to test using actual devices and cloud devices. While testing such applications on cloud-based devices, performing actions like putting different checks in front of device camera for scanning is necessary.
- Fitness applications connect with various BLE peripherals like heart rate monitor, temperature sensor, activity tracker, etc. Apart from Bluetooth connectivity related scenarios, it is important to verify the application's behavior for different test data values e.g. application behavior such as a case of abnormal heart rate. It is difficult to simulate scenarios involving different combinations of BLE peripheral events coupled with varying data for cloud-based devices.

Some applications might be using device's biometric authentication system (i.e. Apple's Touch ID). Testing of such applications may include scenarios like how the app behaves in cases of successful and unsuccessful biometric authentication on devices without having biometric hardware.

## 2.1 Infostretch Mobile Automation Framework

Infostretch has developed a Mobile Automation Framework to help customers automate test scenarios involving device hardware. This solution works for real devices as well as cloud devices. Infostretch Mobile Automation Framework doesn't need application source code. Infostretch Mobile Automation Framework works on an application binary. Also, the framework can accept commands from any automation script and hence can be used in conjunction with any user interface (UI) based test framework, enabling customers to achieve highest level of test automation.

**Solution Overview**

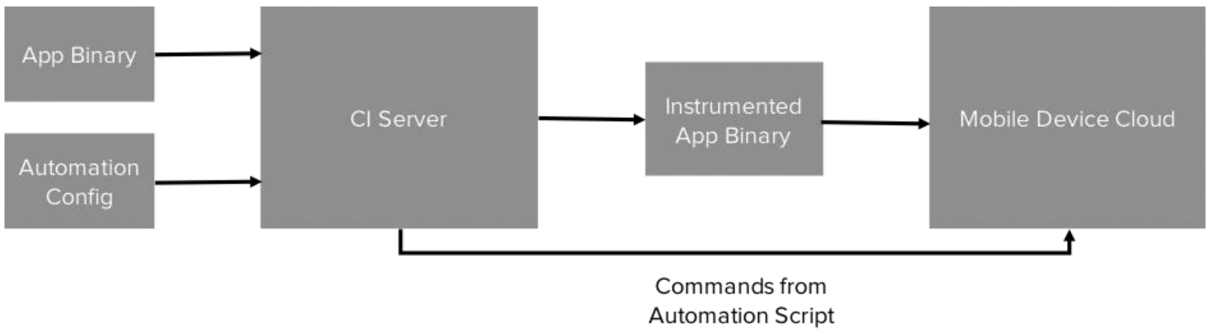

Diagram: Overview of the process flow for integrating and optimizing Mobile Automation

While writing the automation project, the peripherals that are required to be tested using the Mobile Automation Library are specified in the Project Configuration file. Based on the peripherals specified, the continuous integration (CI) server will prepare a customized mobile test automation library. This custom library is then integrated with the application under test. This application is then installed on actual devices or cloud-based devices for test case execution. Once the test suite is triggered, automation script will send command to Mobile Automation Library to simulate different hardware / sensor conditions based on the test case requirement.

Peripherals currently being supported in the Mobile Automation Library are:

- Camera
- Touch ID (Fingerprint-based authentication)
- Apple Pay (Biometric authentication for payments)
- GPS (Location Coordinates)
- BLE (Bluetooth Low Energy)
- System Date & Time

## 2.2 Mobile Automation Architecture

Mobile Automation Library's capability of simulating various hardware conditions is the heart of the framework. This library gets injected in the application binary (source code not required) at runtime, just before the test suite is triggered by CI. The injected library monitors the system calls and messages between the OS & application. The library gets commands from an automation script via REST API or Socket Communication and then changes or spoofs the data or conditions based on the test case execution.

A high-level diagram below explains the working of the solution:

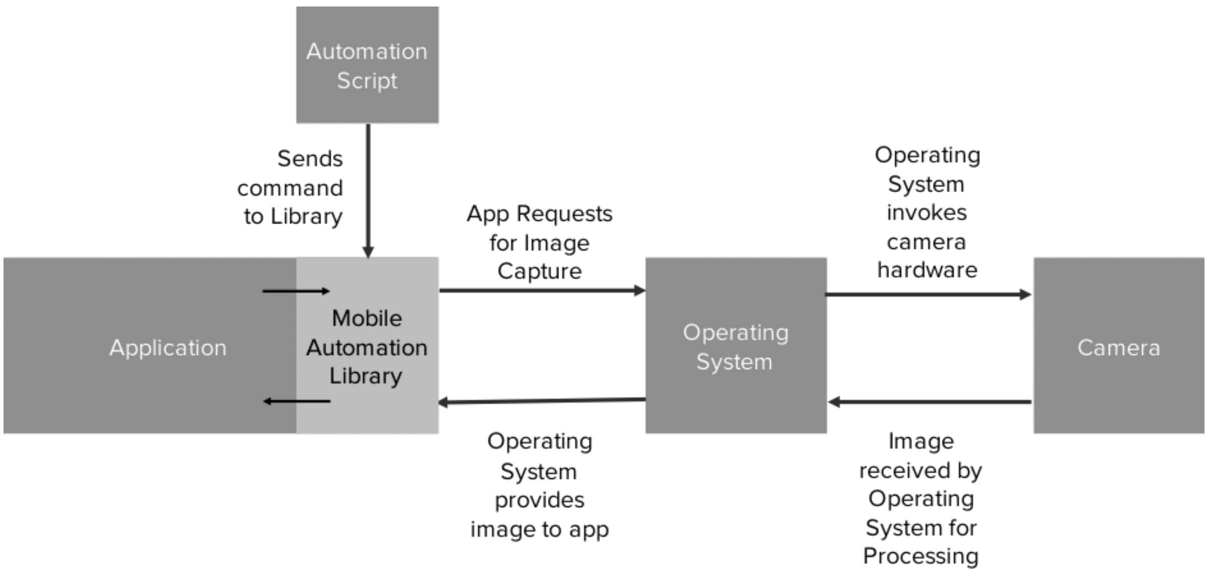
Automation Script
Sends command to Library
App Requests for Image Capture
Operating System invokes camera hardware
Application
Mobile Automation Library
Operating System
Camera
Operating System provides image to app
Image received by Operating System for Processing

Diagram: Application-Hardware Communication Flow

The diagram above depicts how an application requests an image from the camera peripheral and how the request gets fulfilled subsequently. The application requests the operating system to capture an image. The operating system uses camera hardware to capture a picture, process it and then returns it back to the application. Mobile Automation Library constantly monitors the communication between the application and operating system regarding camera capture operation. While running the test automation script, whenever the operating system will capture an image and send it to application under test, the Mobile Automation Library will replace that image with a simulated image. The application will always get our simulated image which will be appropriate for the automation scenario. By using this approach, we can provide images of different checks covering various test data as required by test scenario.

# 3 Automated Bot Testing

## 3.1 Overview

The term “bot” refers to both chatbots as well as voice-based bots. It represents the automated technology that understands user’s queries and then responds accordingly or performs an action. If I am interacting with a financial assistant bot and ask, “What is my net worth today?”, the bot will be able to understand what my intent was and then provide the corresponding figure as the response.

Before we move to testing of bots, we will discuss the common concepts and terminologies required for understanding. In the industry, bots can be classified into three types:

- Chatbots
- Voice Bots
- Hybrid Bots

**Chatbots:** They interact with users through text-based messages. Popular channels of chatbots are Facebook Messenger, Skype, Slack, SMS, etc.

**Voice Bots:** They interact with user using voice-based responses. Popular channels of voice bots are Amazon Echo (Alexa), Siri, Google Home, etc.

**Hybrid Bots:** They react to users’ questions using both text or voice. Very limited number of bots are available in the market that have capabilities of replying using text as well as voice. In near future, industries will see a wider adoption of such bots that allow users to communicate using both types of interactions – text and voice.

Bots cannot only answer questions posed by the users but can also initiate proactive communication (text/voice enabled) based on certain predefined triggers. For example:

- A weather bot would provide you the weather forecast for the day every morning at 6:00 am.
- A finance bot would alert you whenever your stock has reached to your defined level.

## 3.2 Need for Automated Testing of Bots:

**Manual testing is time consuming and costly:** Testing the bot manually is a cumbersome and time-consuming task due to the different flows created in the decision table defined for bot's responses. For example, take a simple bot which takes five steps to finish a transaction. In that decision flow, each step has four different options, and further flow is limited, there would be 256 unique flows which would need to be tested. We already know the limitations of the manual testing as it creates boredom and is often error prone.

**Lack of Automation Framework:** Bot development platforms don't provide automated testing capabilities. Current automation frameworks - e.g. Selenium provides limited options for testing of the bot.

## 3.3 Automated Testing of Bots:

Let's define "What" to test for the bot. We will confine our scope to the chatbot and voice bot only, as the hybrid bot is the mix of these two bots.

Types of testing for bots:
- Core Testing
- Peripheral Testing

**Core Testing:**

While testing any kind of bot - irrespective of its type (chat/voice/hybrid), a few core parameters should be tested to ensure that the bot is able to converse effectively with the users.

**Understanding User Intent:** Success of any bot relies on right understanding on what the user is trying to say. User converses with the bot using natural language, so, by processing the user’s utterances, the bot must be able to understand the user’s intent. So, when the user asks – “who won the Super Bowl game last year?”, the bot should be able to identify that the user’s intent is to “know the winning team".

**Formulating Appropriate Response:** Another important aspect of the core testing is ensuring bot's response is understandable by the users. We need to test the bot’s response when a specific intent has been triggered. So, taking the example above, against the intent of "knowing winning team", the bot’s response must be to return the correct winning team information using natural language. If the user has asked for last year's winning team, the bot should respond with the name of the 2017 Super Bowl Championship winning team. So, we need to verify that the bot's response is in line with the user’s intent.

**Peripheral Testing:**

This testing is specific to the type of bot under test. There would be different parameters to be tested for chatbot vs. voice bot. A list of parameters is shown below in the table. Peripheral testing is equally important as core testing, as it is a significant contributor in ensuring quality around overall user experience while interacting with bots.

## 3.4 Factors to be Tested for Bots:

**Factors to be tested for chatbots:**

1. Different Response – Same Query:
   Smart bots would react differently to the same query. When a user mentions “thanks” it would reply as – “Welcome” or “My Pleasure” or “No problem”
2. The bot’s understanding of Intents:
   Different users ask the same query in different ways. For example, User 1 asks – “Growth of my portfolio” whereas User 2 asks “Percentage change in my portfolio”
3. Typo Errors Understanding:
   How far a bot can understand the typo error from a user without polluting with other intent.
4. Response time from the bot:
   How much time your bot is taking to respond back to your user’s queries. Timeout defined for the bot response must also be aligned to that during automation
5. Multiple Queries in single sentence:
   How does your bot handle multiple queries in single statement? User asks – Show me the suspicious transactions value and total loss in 2017
6. Mixed Languages Query:
   Can your bot understand the multiple languages that has been asked? User may write - Combien avez-vous facturé pour mon POS system?

**Factors to be tested for voice bot:**

1. Different accents, gender:
   How does a bot behave for different accents & gender combinations - American female, British male?
2. Same meaning different utterance:
   Yes, yeah, true, exactly, certainly, etc. can be used interchangeably. The bot must understand them.
3. Different pronunciations:
   People often pronounce “assessory” instead of accessory – does your bot understand the essence of user’s intention?
4. Punctuations:
   How bot interprets the punctuations: Woman, without her man, is helpless – vs – Woman! Without her, man is helpless?
5. Background Noise:
   Check for the effect of noise on the bot’s capability to understand user’s intent.
6. User speaking at distance:
   Effect of user speaking from distance, or in case of listening device being stationary (e.g. Echo) and user is moving and speaking – how does that impact bot’s behavior?

## 3.5 Approaches for Automation of Bot Testing:

**Mimicking the user's action:**

Under this approach, user's actions are mimicked and the bot under test would interact with an automation script that is playing the role of a user. Infostretch’s Automation Testing of Bot Framework has a capability to test chat and voice bot using this approach.

While testing the chatbot, our framework extracts the test data from MS Excel and does the response validation based on the response from the bot under test.

While testing the voice bot, our framework extracts the test data from MS Excel, converts the text to speech, and checks the response from the bot under test. For Peripheral testing, we apply conditioning such as varying distance, generating noise to simulate real-time conditions in our framework or allow user to do conditioning in the test data itself.

**Headless Testing:**

Under this approach, it is required to spoof the response from channel and interact directly with the bot server to understand the response to validate it. Infostretch's Automation Testing of Bot Framework has a capability to spoof the response for chat and voice bots. For test data, we rely on the MS Excel or CSV or allow specific format.

In both approaches – Mimicking or headless, using the framework we can do intent validation and response validation.

For test data, we have utilized MS Excel for flow and test data both. An advanced approach to that is to bifurcate both. Utilizing the mind map diagram for the flow and business rules/logic of the bot, while for the test data, utilizing MS Excel or TDM would serve the purpose. Our framework follows a specific format derived from mind map and creates the test scenarios based on the test data. So, any change in the business logic gets reflected in the Mind map diagrams (which is popularly used for development of a bot) and for the testing of the data, it relies on MS Excel or any test data server.

## Summary

As the adoption of newer technologies like chatbots and voice bots increases, the number of applications built upon these technologies will continue to multiply. Enterprises are increasingly trying to reach out to their customers on these new platforms. Along with bots, the application architecture is also undergoing a huge shift as sensor data collected from peripherals like camera, BLE, GPS is being leveraged to improve the functionality of the mobile applications.

This white paper lays down a fundamental approach to test the evolving mobile application by automating the end-to-end testing cycle of the application using Mobile Automation Library. It allows cloud-based testing of mobile applications by mocking the functionality of the sensors to speed up the testing of applications.

Bot Testing Framework discussed in this white paper lays down a structured approach to identify the key parameters to be tested to ensure that the bots are functioning as per design. It discusses the specific factors to be considered while testing chatbots and voice bots. The white paper also defines different approaches that can be employed for testing of bots. Using the methods discussed here, it is possible to accelerate the testing cycle of bots; thereby shortening the overall time to market for bots.

## References

QMetry Bot Tester: https://www.qmetry.com/qmetry-bot-tester/
QMetry Bot Check: https://www.qmetry.com/qmt-bot-check/