Android Test App
Messaging For In-App SDK

by Rachael Gay - B0063185
Rachael.M.Gay@dal.ca

Performed at:
Salesforce.com, inc
Halifax, NS (Remote)

Saturday, August 29th, 2020

Prepared in partial fulfillment of the requirements of the Faculty of Computer Science Cooperative Education Program

1 Introduction

1.1 Organization

Salesforce is a large Customer Relationship Management company based in San Francisco, California. They employ over 49,000 employees worldwide and thousands more through the "Salesforce ecosystem" as Salesforce Developers and Admins at external companies. They operate under the values of Trust, Customer Success, Innovation, and Equality. They also created the 1-1-1 philanthropy model which encourages companies to donate 1% of their time, 1% of their product, and 1% of their profit to charity.

1.2 Project

My project for the summer was to create an Android app that streamlined the testing process for the Message for In-App Android software development kit. This is an SDK that allows customers to quickly and efficiently add asynchronous customer service chat capability to their mobile apps. It also includes the ability to use push notifications and utilizes persistent conversations which eliminates the problem of having to explain the same issue to more than one agent, as the conversation history is saved.

1.3 Role

My role at Salesforce was a Software Engineering Intern in the Service Cloud, working for the Digital Engagement Team. The expectation of my role this summer was to create the test app from scratch, building out the Android version of the test app which had already been built for iOS.

2 Tasks

*All figures are in an appendix at the end of the document

2.1 UI Design

During the first four weeks of my internship, I worked on getting ramped up on GitEnterprise and Android Studio, taking a course on the Android Jetpack Libraries, and learning the basics of Kotlin and XML. The tasks for this period included building menus with navigation (Figure 1), creating UI elements for configurations such as custom radio buttons (Figure 2), and getting acquainted with model-view-viewmodel architecture. I also began working with Android's shared preferences library in order to save configurations changes made by the user in the settings fragment. This included storing the configuration changes as well as retrieving information from shared preferences in order to update the UI as each button was clicked. For instance as each radio button was clicked, the other radio buttons needed to automatically become "unclicked" which required a UI update.

2.2 Live Data and Room Database

During the middle part of my internship, I began using LiveData which allowed for automatic UI updates. This eliminated the need for use of shared preferences, and allowed for the completion of another task: to create a way to save and load multiple different configurations. I achieved this goal by using the Room Database library, which creates an easy interface between Android and SQL Lite databases. In order to load the profiles, I had to create a RecyclerView which updated live from the database every time a profile was created (Figure 3), and which had clickable items that loaded all configuration settings instantaneously within the same fragment. As well, I created a button which reset all of the configuration settings back to a defined default profile that connects directly to the staging environment. Handling all of this data required the use of two-way data binding within the XML files, as well as the use of a ViewModel to handle transactions of data between the view and the database. This also involved using binding adapters and data validation to ensure that no invalid data slipped into the database (Figure 4). Another way in which I used binding adapters was to bind the language code and translatable text for all localizable strings, as seen in Figure 5. At this time there are no translations other than English available in the SDK, so the image shows both the English translation and the gaps where translations are missing.

2.3 SDK Implementation and Demo

In the final four weeks of my internship, I integrated the test app from my own GitEnterprise repository into the main Android-SDK repository. This required a significant amount of refactoring, and a lot of updates to the dependencies I had been using. Once I had integrated the test app, I was able to implement the SDK into my project, allowing for the display of a chat feed to appear with a menu button click. All of the configuration settings were able to be tied into the SDK's models and the default profile connected to the agent console successfully (Figure 6). Once this was complete, the next step was building out UI components in the SDK in order to display more complex data such as carousel buttons, images, and image links inside the chat feed. I was also given the opportunity to demo my application to the Service Cloud during Futureforce "Demo Days".

3 Problems Encountered

3.1 Architecture

A problem I encountered about midway through the internship was a memory leak created by my misunderstanding of the Android architecture. When transferring from shared preferences to live data and the room database, there was a lot of coupling which needed to be remedied. My mentor, Sarah Young, created a solution and helped me learn more about the overall architecture.

3.2 Git

While I understood the basics of Git, working with Enterprise involves using several passwords, signing commits, squashing commits, and many more slightly more advanced

features. I had a few instances of difficulty with Git and was able to take a Udemy course about Git which helped me become much more confident. As well, a teammate recommended two "GitGames" which are command line games that involve searching through branches and files in order to find clues while teaching both basic and more advanced git concepts.

4 Figures

Figure 1: Main Menu

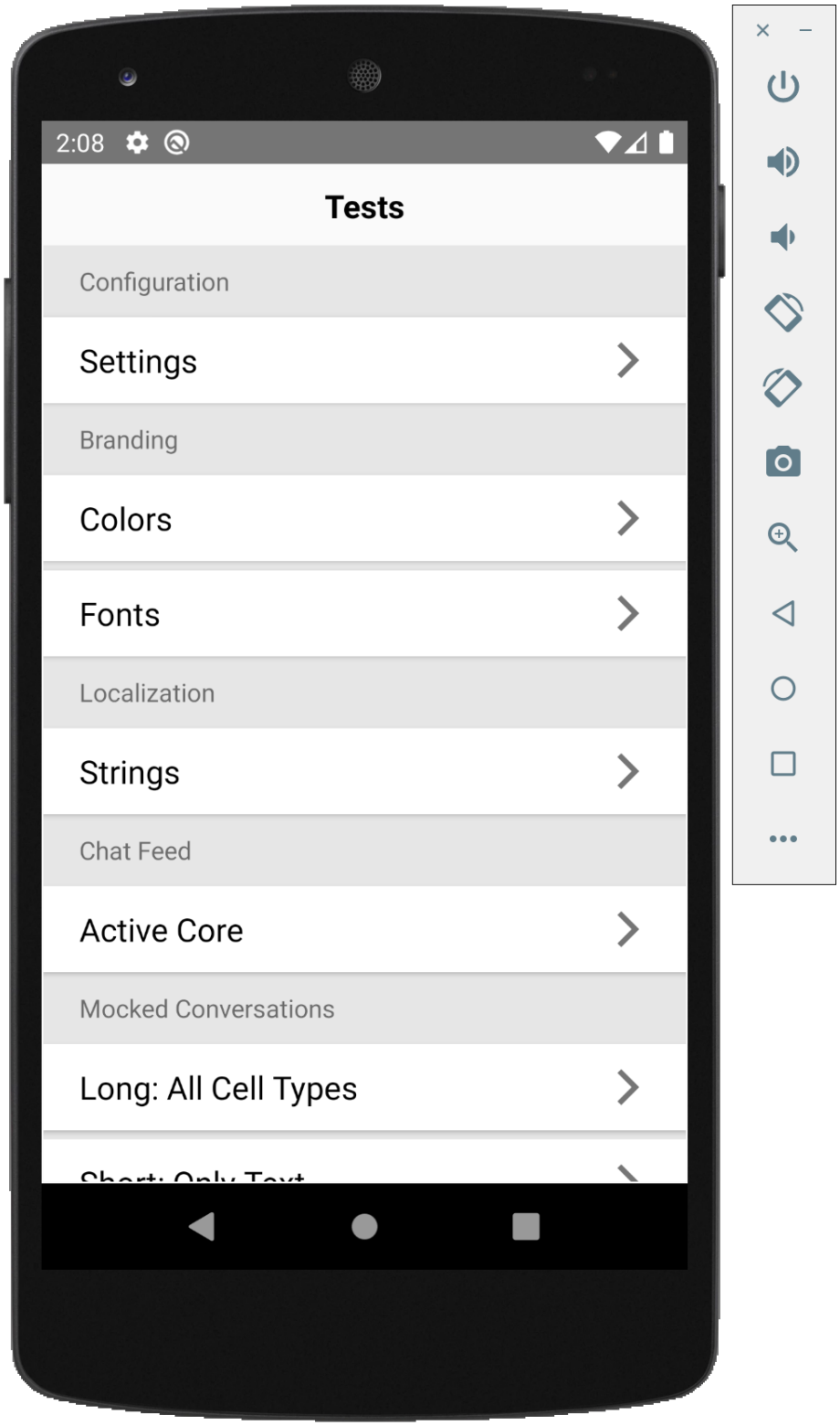

Figure 2: Custom Built Radio Buttons

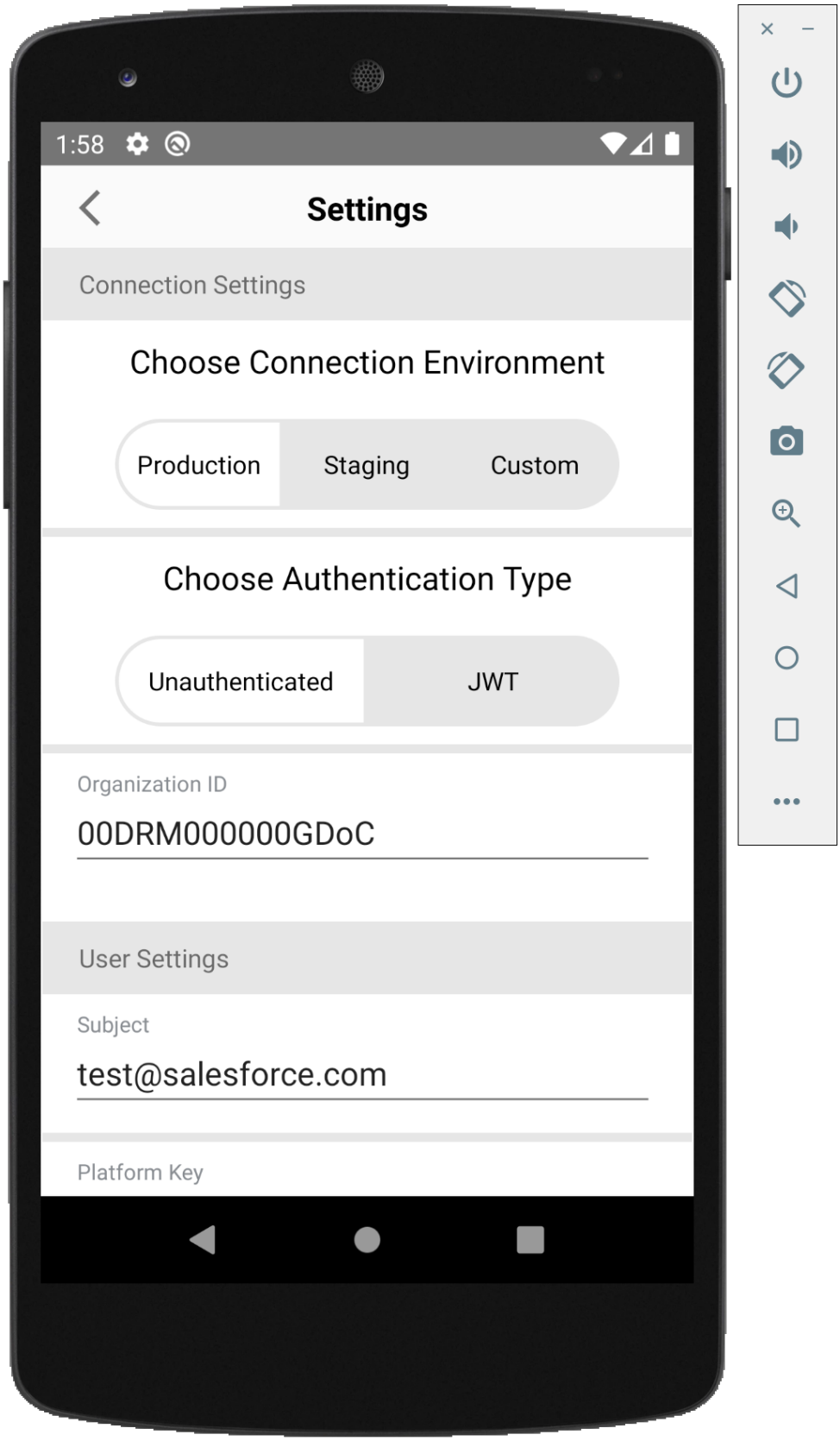

Figure 3: Saved Profile RecyclerView

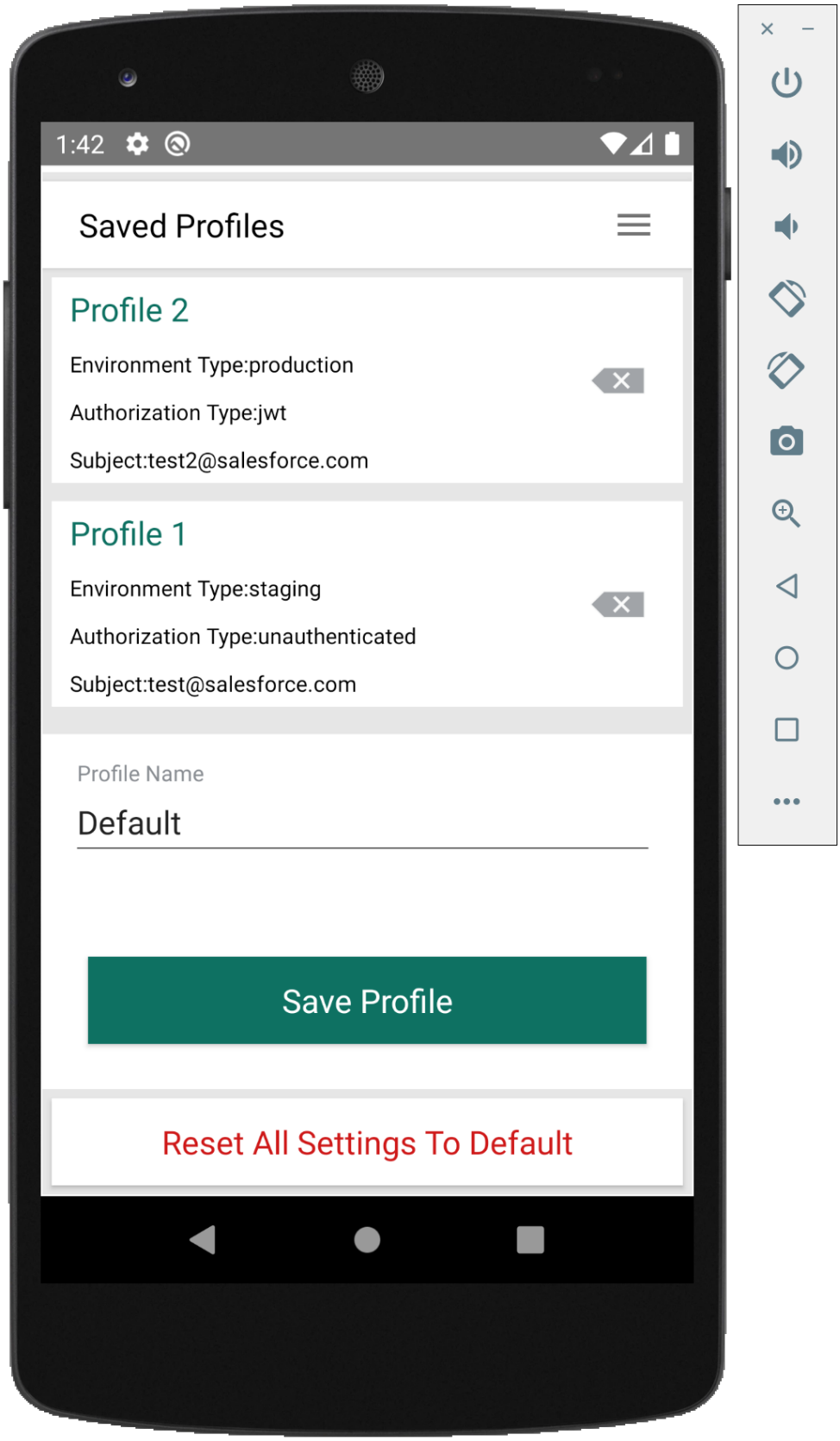

Figure 4: Data Validation

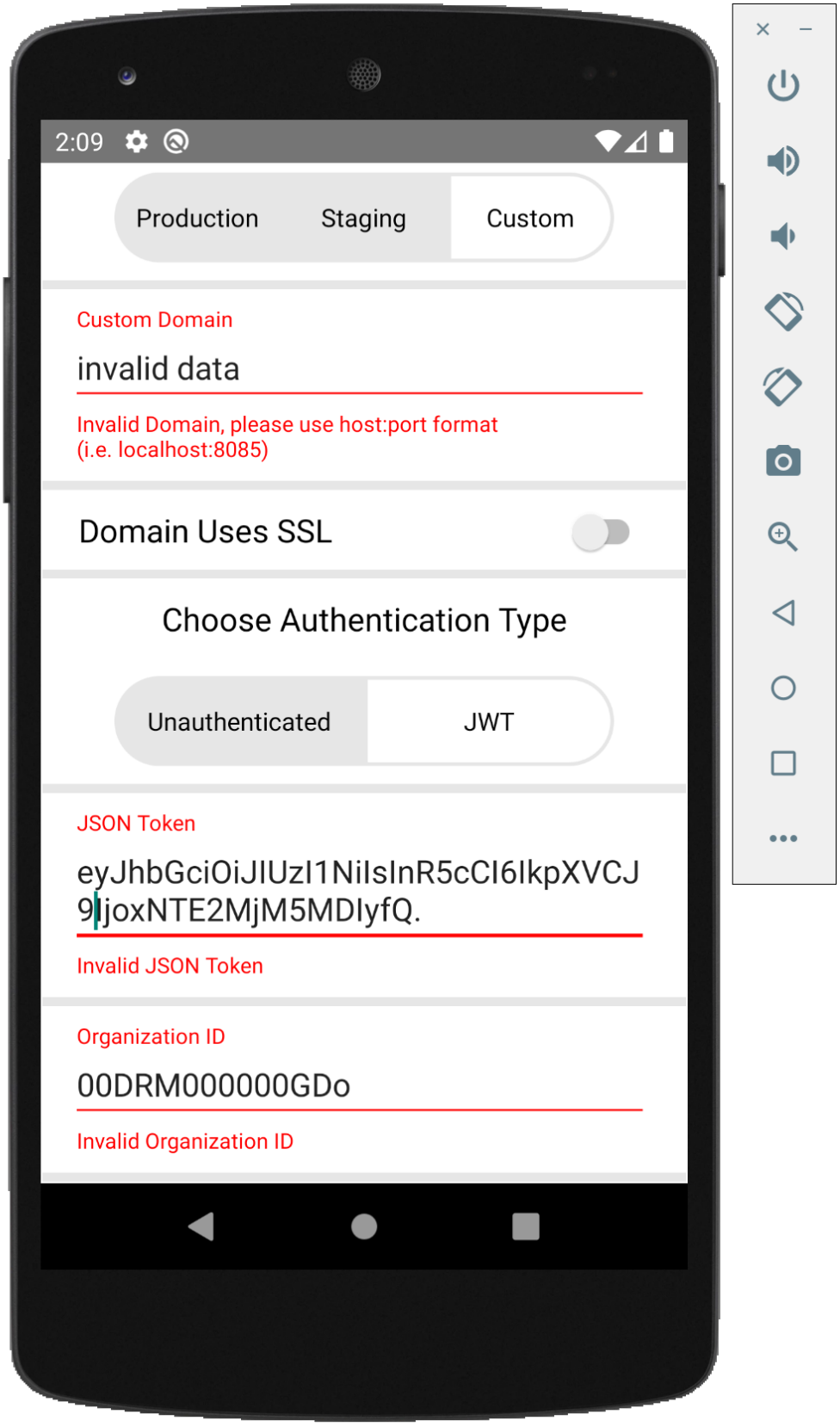

Figure 5: Localization

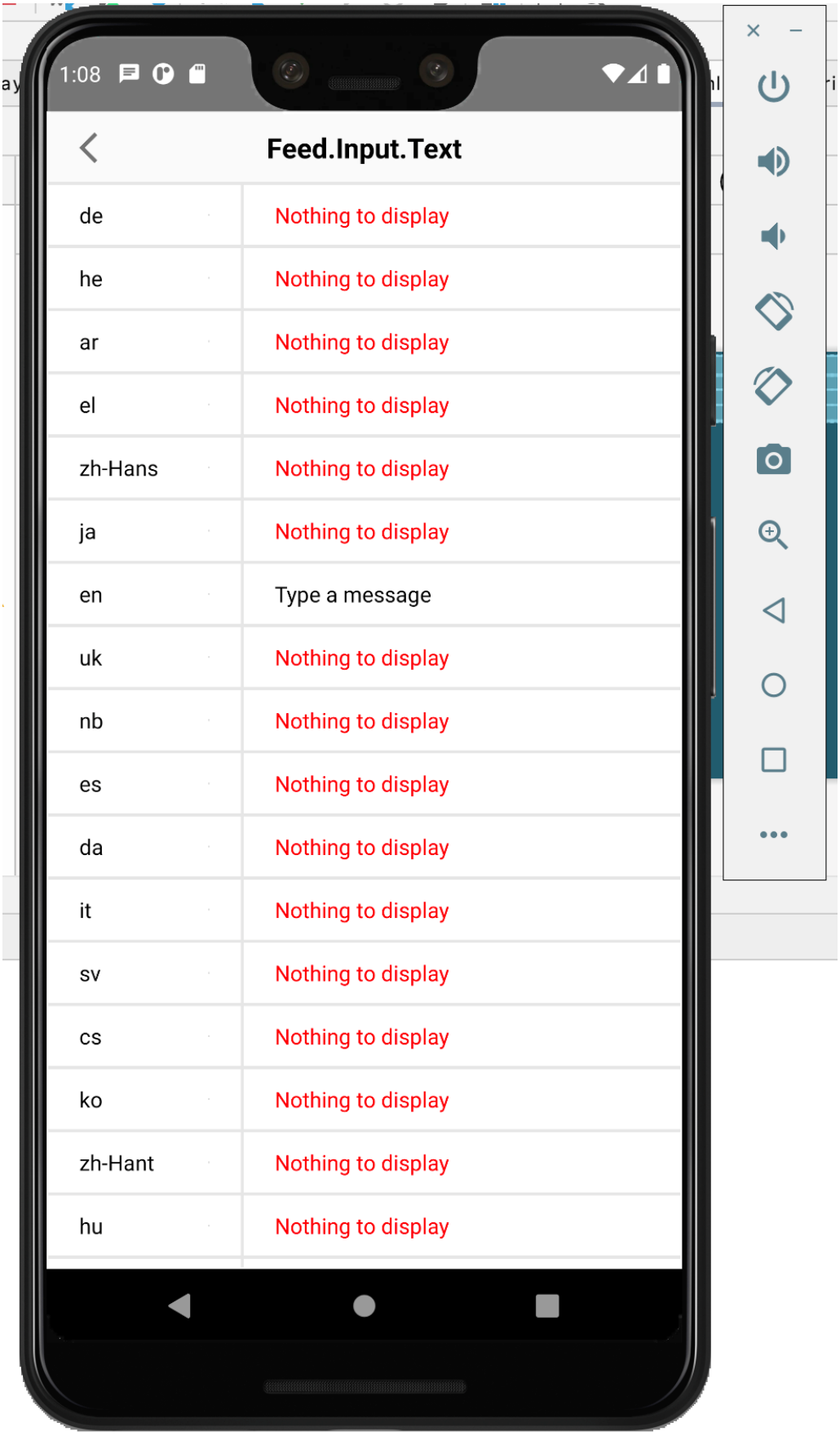

Figure 6: Agent Console