Smart Prompter Technical Overview Paper [Spring 2020]

By: Likhon Gomes (Leo), Dr Chiu Tan, Sarah Lehman

May 2nd, 2020

Introduction

The goal of this app is to harness the power of technology to help older adults with cognitive disability to perform their daily essential tasks. This app works as a reminder that reminds them to perform tasks at certain times to keep them in sync with their daily. This app comprises of two components separated into two apps. One is the admin app, that is made for a caretaker to set alarms for their patient and the other app is the Caretaker app that fetches the set alarms from Admin app and set alarm to remind the patient to complete tasks.

Background

As we grow older, we tend to lose some of our cognitive abilities, one of the most common loss is our memory. Older adults tend to be more forgetful and have hard time to remember their daily essential activities. Tasks such as taking meds, watering plants, feeding their pet, etc. That's why it's hard for them to stick to their goals. That's where care takers come in this situation. Care takers take responsibility of the older adult and get them through the day by reminding them to do their essential tasks. That is why the Smart Prompter research team has taken a unique approach to harness the power of technology to help older adults with cognitive disability to perform their daily essential tasks, without requiring the presence of caretaker on premise. This app is an ongoing research studying the effectiveness of technology and how patients respond to task reminders digitally. Through this app, care takers set specific tasks as reminder on the patient's phone either remotely or on premise. Then the patient is required to respond to the alarm and complete the task. In this research we are researching the reactions of the patient, how effectively the alarm reminds them about the task, and specific side effects from it.

In fall 2019, I worked on laying the foundation of two of these iOS apps and then build them up. By the end of the semester I had two of the working copies of the iOS app both the admin and patient apps capable of doing some of the essential tasks required to do. Tasks such as setting alarms, pushing notification and completing the alarm. Even though the apps were ready, there were a lot of difficulties that hindered the progress of the app and infested them with bugs and inaccuracies. Most of these difficulties came from the very nature of how Apple treats their iOS apps. More about these difficulties have been discussed in the Difficulties section.

In Spring of 2020 when I came back to the research, I worked on fixing some of these bugs and make the app bug free and add some added features. Features such as adding custom voices ring as an alarm sound and showing the user their completed task progress and reward them with a meme was added during this period. I have also added the capability for the caretaker to upload custom images from their gallery or their phone to provide the patient with hints about the task. So far, they are working well.

Challenges

Both android and iOS have difficulties in their own way. While Android developers face difficulties with fragmentation and wide range of device supports, meanwhile iOS developers face difficulties with extreme security measures apples takes to protect their user's data. I have face so many challenges and fixed most of them. Below, I have provided 4 of the most challenging tasks I face in this project that took me more than a day or even weeks to figure out a solution.

1. Share alarm data between two apps

While developing this app I had similar issue where I was not able to share a common database between two apps, which is very much possible in android. Because all of the apps are sandboxed into their own unique shell, they get their virtual resources which can't be accessed by other apps, or at least not easily. Therefore to solve this problem I had to resort to Firebase to send data between Care taker and Patient app. Although firebase solved the problem of syncing the data between apps, it did create another difficulty which was consistency. Because firebase is a remote database in the cloud, it's very hard to keep track of which files have been read and which files have been removed from either of the app. For example, I had a problem when I was trying to implement delete feature for alarms. When an alarm has been created by the caretaker to be pushed to the patient app the alarm data goes to the firebase and sits there waiting to be pulled by the patient app. The alarm can only be pulled in the patients app only when the patient will open the app to check it, only then the app will pull the alarm and request the OS to set and alarm at a certain time. Now the problem arises if the caretaker decides to change their mind and remove the alarm. When the care taker taps the remove button in the alarm, a delete request is sent to firebase instead of removing the alarm altogether. Then the delete request waits in the database to be pulled by the patient app. If the patient opens the app before the alarm goes off, then the app will pull the delete request from the database and request the OS to remove the alarm from the schedule, only then the patient app prompts to remove the alarm from the database. Now, if the patient fails to open the app before the alarm's scheduled time, then the alarm will go off without knowing that it was not supposed to go off. Here you can see it is really terrible way of handling the alarm situation which can be improved, but I will talk about that in the results and discussion section.

2. Setting Notification for Alarms

Some other challenges in this project were, scheduling the alarm in the OS by alarm. There are two types of alarm API iOS provides their developers, one is Local Notification and the other one is Remote Notification. Remote Notification is the usual notification we see in our app when we receive an email, messages or from any other app that requires internet, remote notification is handled in a remote server and pushed instantly because the app doesn't require to run in the background to push the notification, instead it's pushed via the server over the network. Now, in this app I have used Local Notification which doesn't require a server or any other type of infrastructure to push it. This is the type of notification we see when set and alarm, a reminder. There are two types of LocalNotification one is Interval Notification, which goes off after certain period of time and can be repeated. Other one is CalendarNotification which goes off at a certain date/time. For this app I have used Calendar Notification to push the notification at a certain date. Therefore, the app requests the OS to set 5 alarms at a time at a given interval of 1 minutes to notify the user. Now this is quite a bit mess because once the user completes an alarm, then the all the other scheduled alarms should be removed the schedule, otherwise those alarm will still go off.

This whole problem could be solved if we could run code in the background when the first notification is pushed. My plan was to schedule Interval Notificaiton instead of 5 notifications for a single alarm. But the problem is, the OS won't allow the app to execute any code until the app is running in the foreground. Therefore it's is possible only if the user opens the app when they get the notification. But if they don't then the app never gets to execute the code to schedule the alarm, that's why it's not a viable solution and I had to go down a complicated path, which works but not the best way to do it.

3. **Showing patients alarm completion progress**

This semester I worked on adding a reward feature where the user is shown a progress bar showing them how much task is done on that single day and reward them with a funny meme. Now, there's a caveat to that. The progress bar only accounts for the tasks done for that one particular day. It will not work for leftover tasks from another day. The reason being, it's tricky to account for all of the active alarms because right now the database is storing all the alarms including active, recently completed and zombie alarms (that has been completed a long time ago but never removed). The zombie alarms are still available in the database because I did not write a function to remove them from the app of in the database because we will be keeping the alarm data for study purposes. Accidentally removing the alarm will destroy the whole study. So, it's safe to just keep them there and remove them manually from firebase when the data is harvested safely.

4. **Set custom voice recording as alarm tone**

In addition, another feature I have worked on this semester was the ability to set custom alarm tones. This was an extremely tricky task because iOS limits what you can do, but deep inside, I knew this is possible because I used an app named "zello" which has implemented this feature before. Right now, the custom alarm feature is working well, with some minor exceptions. When the user records his/her voice, the playback is played via the earpiece speaker of the phone (I have no idea why, but the fix should not be hard). So, the user might think the sound is too low, but when the alarm goes off in patient's phone, it's pretty loud and clear.

Design

I have designed the app from the ground up using the Android app as an inspiration, although the iOS app does not look exactly similar to the Android app, but the core function should be similar. For maintainability, I have adopted RootViewController approach to unify the app's interface, below I have discussed little bit about the architecture of the app and the database.

App Architecture
At first the app's view controllers were built by inheriting UIViewController. As the app grew bigger, it became harder and much more complex to maintain the UI consistency. So I have created a RootViewController that defines a specific layout that can be inherited in most of the view controller that look similar. Some of the layout laid out in the RootViewController are, topLeftButton, topRightCornerButton, ViewControllerName and a Button stack towards the bottom. Most of these functions have been laid out in the RootViewController so my successor will only need to add the elements in the place holder without worrying about the size and location of the button. The placeholder will take care of that and keep it consistent. I planned to do similar with patient app, but ended up not doing it because the patient app doesn't have enough view controller to implement this.

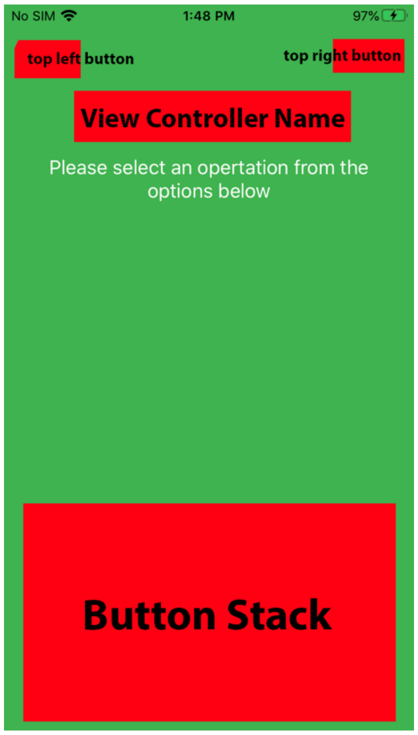

Fig: A diagram of the root view controller

The Database
Right now, both of these apps are using Firebase as a medium to sync data between the admin app and the patient app. It is working but there are certain limitation with firebase such as maintaining zombie alarm in firebase, maintaining the status of the alarm without an iOS app. These all could be done using cloud functions which is another hassle to deal with.

In the future if both of these apps are merged into one, then the transition should be smoother than expected because I had already implemented an on device SQL database called GRDB. I had written most of the functions for basic tasks for Fetch/Update/Insert etc… although they were not completed, I believe my existing code will help guide my successor if they choose to retain the database. More documentation about GRDB can be found here: https://github.com/groue/GRDB.swift. Also, I just found this recently, if the team wants to retain the app in two separate state but exclude firebase, that is also possible using containerURL: https://developer.apple.com/documentation/foundation/filemanager/1412643-containerurl

Results & Discussion

As of now the app is capable of doing most of the tasks. The caretake is capable of creating an alarm, deleting and alarm, set custom alarm image to provide the patient with hints about the task to be completed, keeping track of complete and incomplete alarms. The caretaker can also setup custom voice as an alarm tone for the patient. On the patient's side, the patient app can download the alarm, set the alarm to go off at a certain time. It also allows the patient to upload a picture of the task they completed.

Now even though the app is capable of doing most of the tasks, the app is still vulnerable to potential bugs and inconsistencies. Over the last couple of week I was requested by Tania and Katherine to make some fix some bugs and provide some improvements. Some of their bug fix requests has been mentioned below. I was asked to add color indication on the slider to let the user know which way to slide (added red for the left side of the slider indicating delay and green on the right side, indicating completion), make changes to completion page on admin app (fixed the typos and keeping the data consistent), remove seconds from the alarm time, remove alarms that have already been completed (solved by removing alarms from the OS once they are completed and even if they are delayed), adding UI consistency for Notched iPhones (fixed by adding SafeAreaLayout guide in the constraints). I have fixed the issues on the following revisions, and there were some other inconsistencies they talked about, and they may still exist in the app. These inconsistencies will be solved when both of the apps will merge in the future. Then every single update can be done in Realtime on device therefore eliminating the necessity to open up the app to pull contents from firebase.

I have already implemented the basic foundation for on device SQL database, so it should be easy to implement. Although merging both apps into one will solve the inconsistency problem, it will likely not address the local notification issue. Local Notification problem can only be solved by moving the backend logic to a server and then implementing Remote Notification using Apple Push Notification Service.

Appendix

Side Notes

- All of the data base data are store in Database section in firbase
- Pictures/audio sounds are store in Storage section of Firebase
- The app will crash if the database is inconsistent/ so make sure the whole database is cleaned properly when zombie alarms are removed

How to Install the app, go to this github link : https://github.com/likhongomes/Smart-Prompter-iOS and git clone the repository and open it using xcode. click the run button or press cmd+R on keyboard to run the app, on a simulator or your iPhone.

Requirement: iOS 10 or above and xcode version 10 and above.

Technical Documentation:
PDF for has been included with the folder
Technical Documentation for Admin App: SmartPrompter-Admin_TechnicalDoc.pdf
Technical Documentation for Patient App: SmartPrompter-Patient_TechnicalDoc.pdf

Online Documentation: https://likhongomes.github.io/Smart-Prompter-iOS/

Also, these documents are available on the Github Repo.