ON THE SPECTRUM: CONTACTING EMERGENCY SERVICES FOR NONVERBAL AUTISTIC PEOPLE

DETAILED DESIGN

October 22, 2017

Team 7103

Vidya Iyer, Arsh Momin, Ecclesia Morain, Frances Tsenn, Ethan Vargas

Table of Contents

List of Tables

List of Figures

1. Introduction

Our client, Matthew Bollinger, would like to address the difficulties that nonverbal autistic people have when calling emergency services through the development of a mobile application called “On the Spectrum.” This application will aid users in contacting and communicating with emergency services, which will allow first responders to act without confusion or delay. The Center for Disease Control (CDC) estimates that one in 88 children have Autism Spectrum Disorder, and most have difficulty communicating effectively with language [1]. In an emergency situation, people living with Autism Spectrum Disorder may have difficulty communicating relevant information to others. Additionally, the lack of responsiveness and atypical behavior may cause first responders to misinterpret the situation. Many have limited training on autism spectrum disorder and often do not know how to approach or assist individuals with autism, which may cause the problems to escalate [2].

“On the Spectrum” will help nonverbal autistic users communicate with emergency services by using user information and situation descriptions when making a call. In preparation for emergency situations, the application will allow the user and the user’s caretaker to input personal information that would be relevant to emergency services in times of crisis, such as the user’s name and location. During an emergency, the user will select an option that describes the type of emergency they are in and a call will be made to emergency services. Once a call is made, the application will repeat the necessary information so that first responders can act immediately.

Our application will be implemented on the Android platform. To initiate calls to emergency service providers and share information relevant to the emergency, our application makes use of Sinch, a third-party Voice-Over-IP service provider, which will host the call and relay the important information to emergency service providers using a text-to-voice feature. Additional major components of our application include: text-to-voice translation in real-time, logging previously made calls, and allowing parental controls. In order to ensure that all of the user’s health information is kept private, all information, such as customized profile and emergency information, is saved locally on the user’s device. Throughout the application, we made user interface design choices that would make it easier for users who are on the spectrum. For example, we decided to use pictorial buttons, a form of nonverbal communication, which makes our application more intuitive to our target user group.

This document will detail the architectural solution that our team chose in creating this application. The System Architecture Design provided in Section 2 describes how components of the application interact with each other and shows the control flow of the application given a specific scenario. The Data Storage Design provided in Section 3 describes how application data is stored and shows an ER diagram. The Component Detailed Design provided in Section 4 provides specific static and dynamic component diagrams in order to explain the operation of the application’s common processes. The UI Design provided in Section 5 showcases screenshots of the application’s user interface.

2. System Architecture

In order to understand the structure of our application we have provided designs of architecture of our system in Figures 2.1 and 2.2. These diagrams model a high-level description of the structural components of the application and relations between them. The application is defined by three core functional components: the Key-Value store, where the user information is stored, the Sinch library which provides the VoIP service, and the app that the user interfaces with to initiate the app functionality.

2.1. Static System Architecture

The static architectural design shown in Figure 2.1 provides a functional view of the system. In this diagram we show how the components are interrelated and their relation to one another.

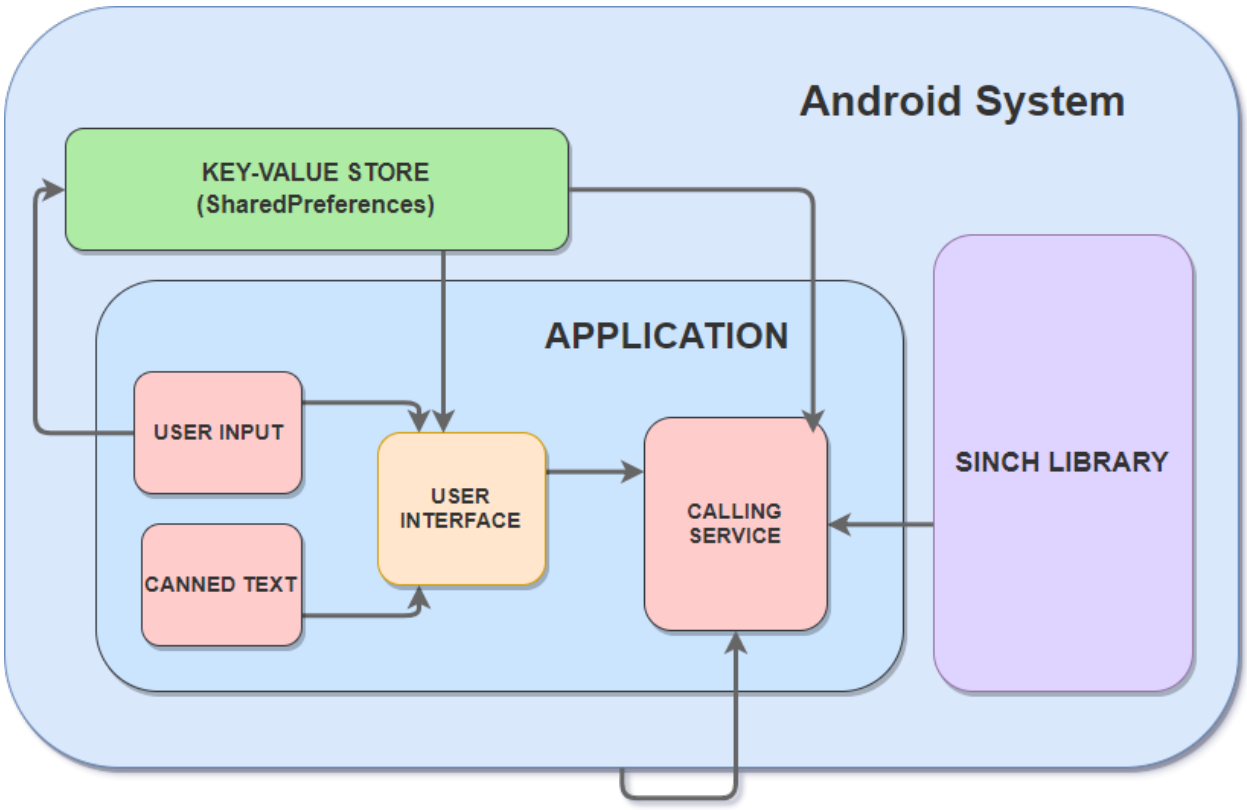

Figure 2.1. Static System Architecture Design for On the Spectrum

Everything that our application manages is contained within the app which allows for a simpler architectural model. At the front end is the user interface through which most control flow is initiated. The user interface is the means by which the app user can make adjustments to the app, such as changing the app's appearance and editing the information that that is stored on the app. The information that the user adds to the app is saved on the device in the key-value store. The Android API specifically references this store as “SharedPreferences” so the naming convention is indicated in parentheses within the diagram."

The application is developed with its own canned text, which is generic information that can be sent to the calling service such that the call still is effective in communicating that a person is in an emergency. This information exists as default information that will be delivered to the calling service if the user does not change the text.

The calling service captures all of the functionality of the app that manages the call to emergency services. This includes initiating, communicating, and ending the call. The call is designed to rely on VoIP as the primary resource to manage connecting the app to emergency services, which is provided by the Sinch library, however if that fails the calling service will look to the Android System to manage the call connection. The Android System describes the specific Android device that the app is downloaded on along with permissions for this specific device such as access to the device's calling functionality.

2.2. Dynamic System Architecture

The dynamic architecture design shows the control flow within the system and the interactions between components of the system as information is exchanged. Figure 2.2 depicts the dynamic view of our application's architecture in a scenario where a user has opened the "Choose Emergency" screen and presses an emergency scenario button to make a call to emergency services.

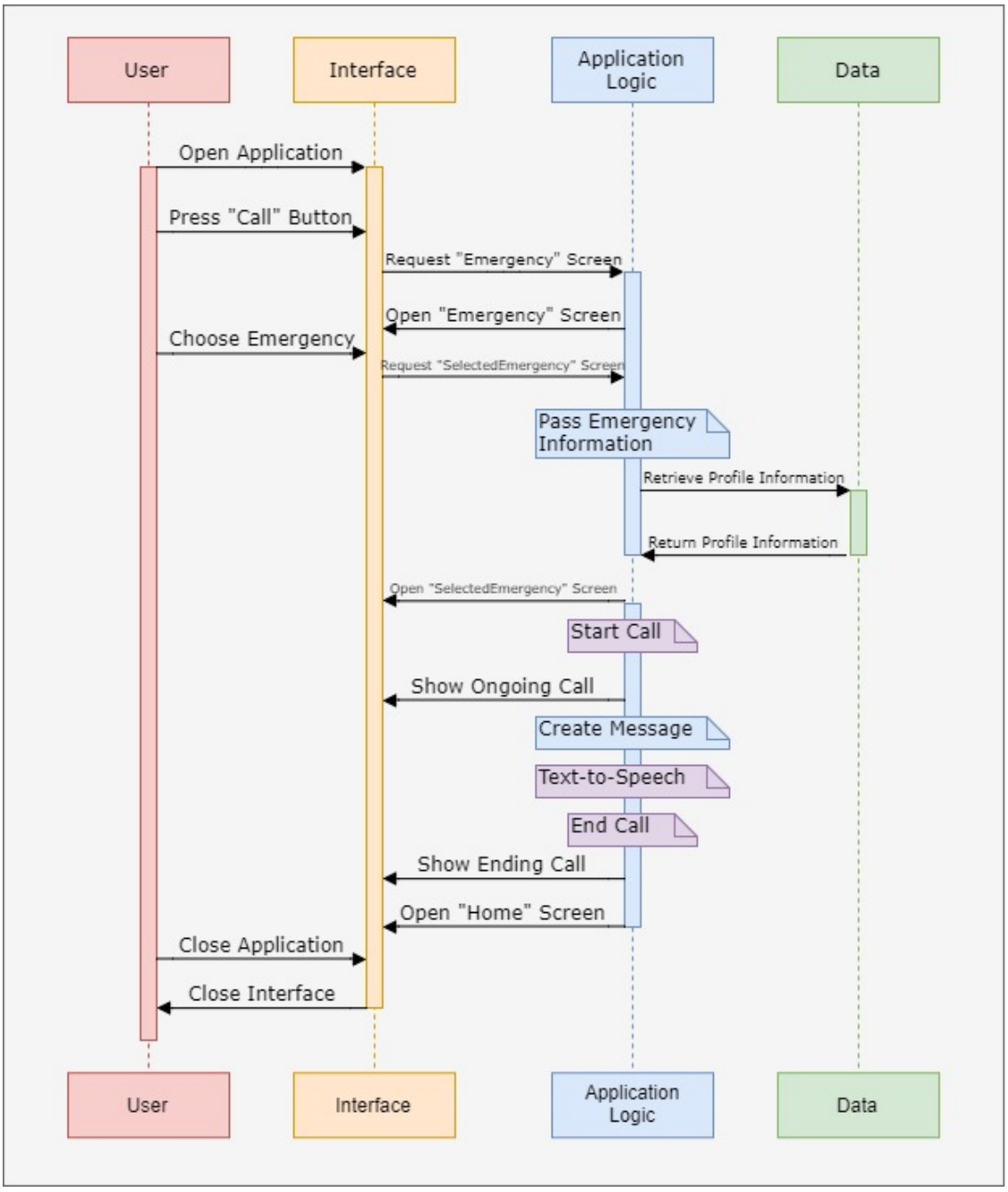

Figure 2.2. System Sequence Diagram for Making a Call

3. Data Storage Design

To address the potential privacy concerns of an autistic individual, the application will not save any information to external databases. Instead, the user's profile information is stored locally on the Android device using the SharedPreferences data object inherent to Android. This data-storage object is made up of a small collection of key-value pairs. The necessary data is loaded when the user navigates to the appropriate screen and is re-saved when the user has edited information.

Table 1 displays an example of the profile information stored in SharedPreferences. Firstly, the fields included in the profile are stored as a series of strings concatenated with ";;", which is unlikely to be used within the actual information. Each of the fields is associated with its own value.

The application also uses a combination of SharedPreferences and the device's SD card to store the emergency scenarios. The images or icons that are associated with scenarios are stored on the SD card under the "/saved_images" folder. The file path to the images are stored within SharedPreferences. Table 2 includes a sample of what may be included in SharedPreferences.

Table 1. Key-Value Pairs within SharedPreferences Relevant to the Profile

Key	*Value*
ProfileFields	Name;;Gender;;Age;;
Name	George P. Burdell
Gender	Male
Age	20

Table 2. Key-Value Pairs within SharedPreferences Relevant to the Emergency Scenarios

Key	*Value*
ScenarioNames	Break In;;Choking;;Lost;;
Break In	/data/user/0/com.ots.tdd.onthespectrum/files/saved_images/breakIn.jpg
Choking	/data/user/0/com.ots.tdd.onthespectrum/files/saved_images/choking.jpg
Lost	/data/user/0/com.ots.tdd.onthespectrum/files/saved_images/lost.jpg

4. Component Detailed Design

4.1. Static Component Design

The major components of the application and the relationships between them are illustrated in Figure 4.1. Here, components represent either the screens of the application or backing systems that provide functionality that the screens rely on. This diagram was developed using the UML 2 standard, which uses the lollipop and socket notation, for which the lollipop- the full circle extending from a component -represents a component that is created or provided by the component with the half-circle.

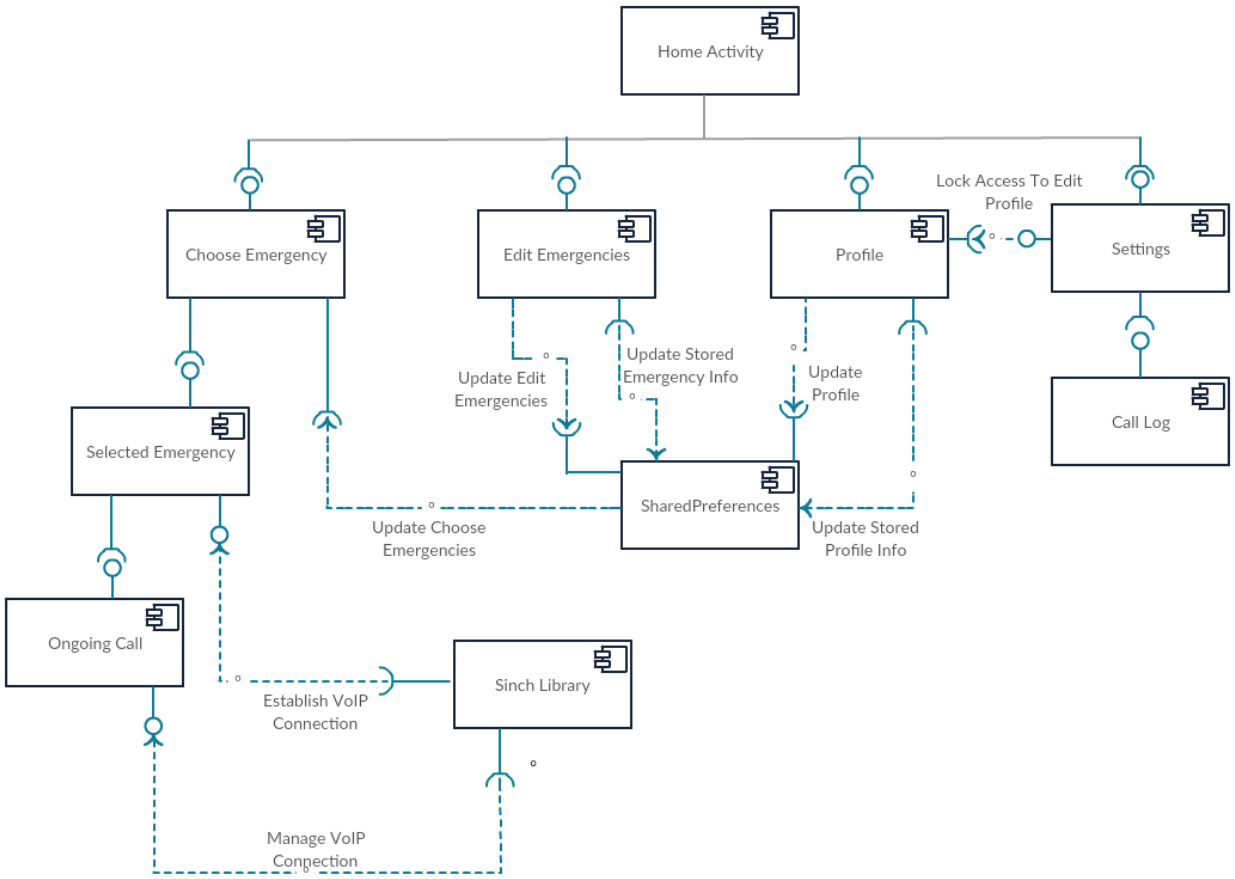

Figure 4.1. Static Component Diagram of the Application

The home activity can lead to one of four other activities: “Choose Emergency” to be used in the case of an emergency, “Edit Emergencies”, “Profile”, and Settings. The diagram is subject to change depending on the success of interfacing with the Sinch library. If the Sinch API, which will both call emergency services and provide text-to-speech over the call, is not compatible with the application, an alternative method of calling and conveying information will be used. The call will be put on speaker phone, and the device itself will perform text-to-speech at the loudest possible volume. This will only affect the part of the diagram that references the Sinch library.

4.2. Dynamic Component Design

The dynamic component design is detailed in Figure 4.2. The dynamic version of the component diagram better characterizes how the application will behave in use. It elaborates on the conditions necessary for interactions between components to be executed.

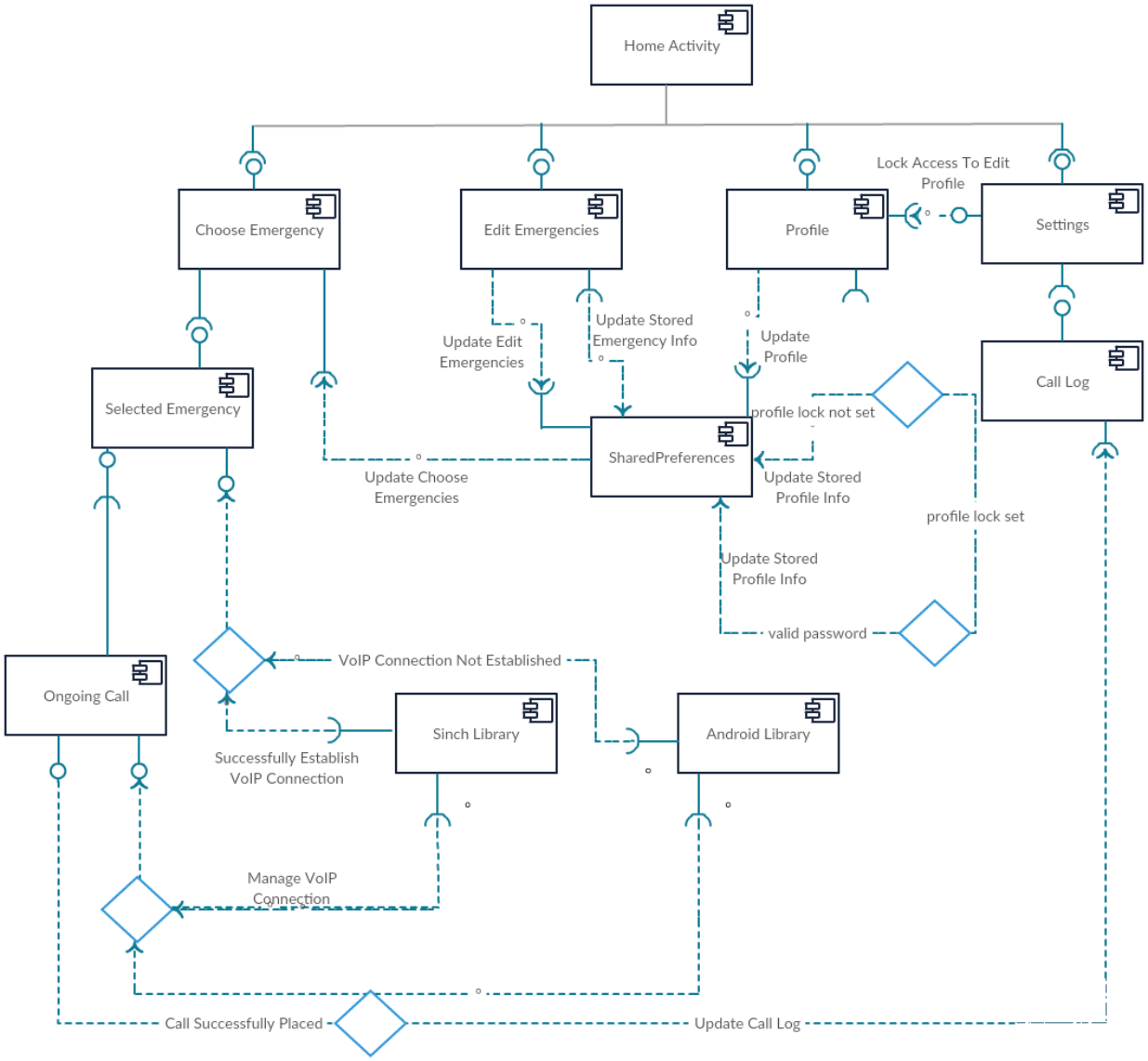

Figure 4.2. Dynamic Component Diagram of the Application

As mentioned previously, the Sinch library may not be utilized. In the case in which Sinch is not used, the Sinch Library component of Figure 4.2 will be disregarded. Instead, the application will always rely on the Android Library when placing the call.

5. UI Design

In this section we present the User Interface (UI) of the mobile app, which users will use when faced with an emergency situation that requires the help of emergency services. We discuss our UI design decisions and describe all major screens that the user will interact with.

When the user opens the application, they will be brought to the home screen shown in Figure 5.1. The screen has a simple layout so that our primary users, nonverbal autistic people, will not be confused with too many design elements or options on the home screen. The emergency call button (located at the top of the screen) is larger than the others because we want the user to be able to quickly identify and press the emergency button when they are in an emergency situation.

When the user presses the “Profile” button shown on the home screen, the screen shown in Figure 5.2 is displayed. This screen contains the personal information about the user that a first responder would need to know in order to provide help in an emergency. These fields are all editable and are filled in with default information when the application is first loaded. A user also has the option of adding additional fields of information.

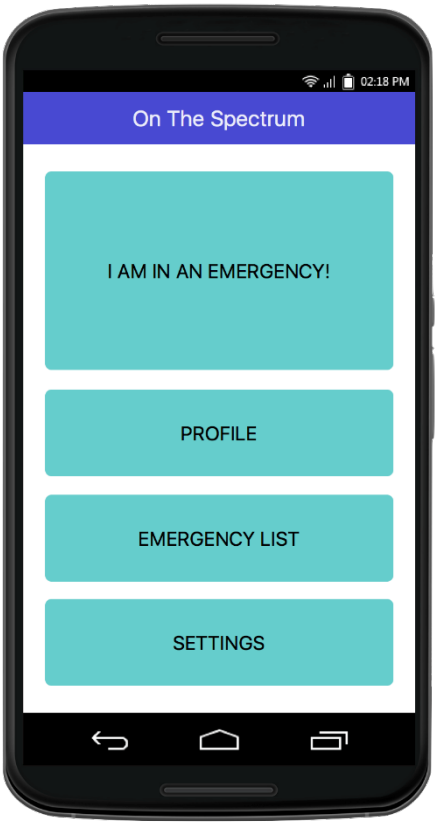

Figure 5.1. Home UI

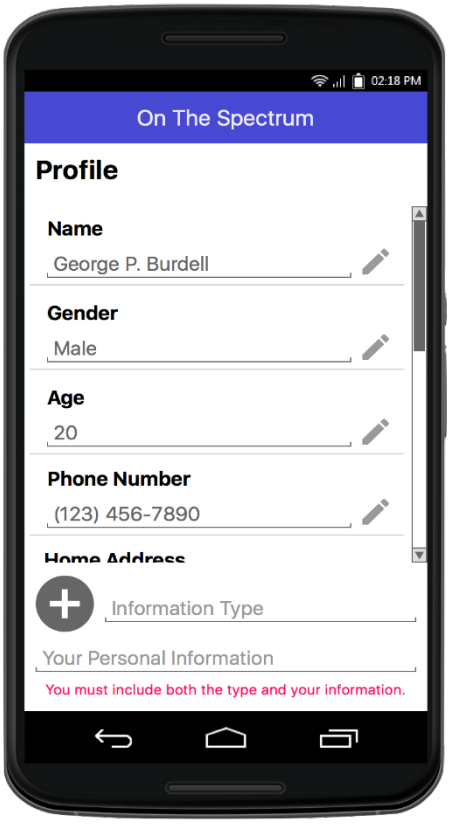

Figure 5.2. Profile UI

When the user presses the “Emergency List” button shown on the home screen, the screen shown in Figure 5.3 is displayed, which shows a pre-populated list of emergencies that a user could find themselves in. By default, the list will include common emergencies, such as a break in, choking, fire, etc. This list will be editable, which is indicated by both the title and the action in the top right corner of the screen. It has large, low-saturated images in order to provide more context for each

situation. The user will be able to modify the list by adding, removing, or editing emergencies depending on their personal situation.

A similar screen, shown in Figure 5.4, will be displayed when the user selects the “I am in an emergency!” button shown on the home screen. The two screens are distinct because, in case of an emergency, there should be no additional functionality that would cause confusion when the user wants to contact emergency services. The similarities were consciously chosen so that a user would have some familiarity with the UI when in an emergency. However, the images are more saturated, which is an indicator to the user that this screen is not for merely editing.

Figure 5.3. Edit Emergency List UI

Figure 5.4. Select Emergency UI

Figure 5.5 shows the settings screen, which is navigated to when the user presses the “Settings” button shown on the home screen. The user will have the option to customize the application’s settings, such as its font, color scheme, and lock. When the user selects an emergency shown in the select emergency screen (Figure 5.4), the screen shown in Figure 5.6 is displayed, which combines the information provided in the profile screen (Figure 5.2) and the emergency list screen (Figure 5.3). This screen takes the associated information and displays the script that will be read when the call is placed. Profile fields that have not yet been filled out will not be included in the

script. When the user presses the “Call” button, the screen shown in Figure 5.7 is displayed and a call is made to the emergency services. From this screen, the user can end the call at any time by pressing the red button.

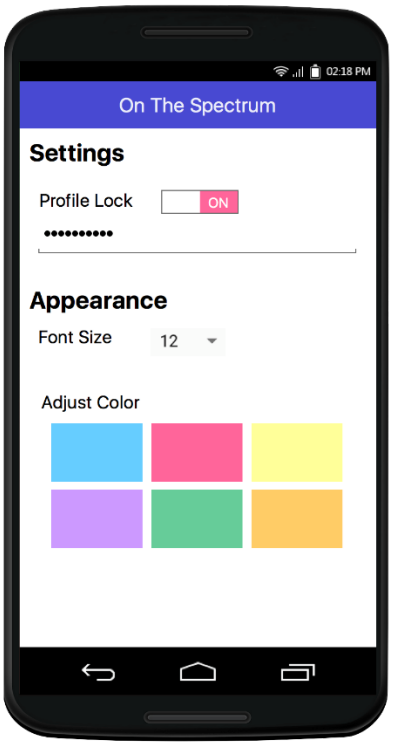

Figure 5.5. Settings UI

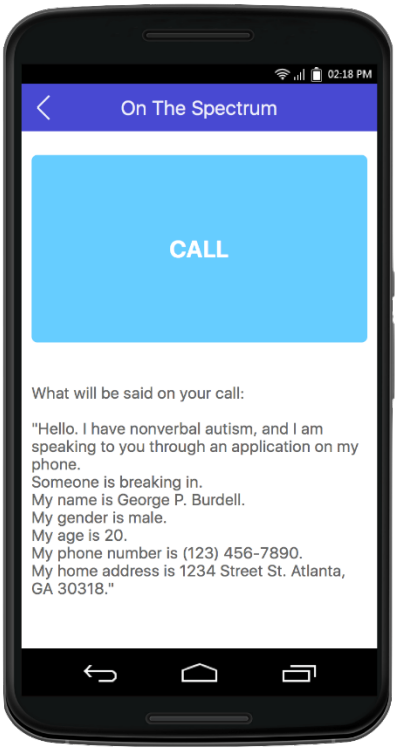

Figure 5.6. Pre-Call UI

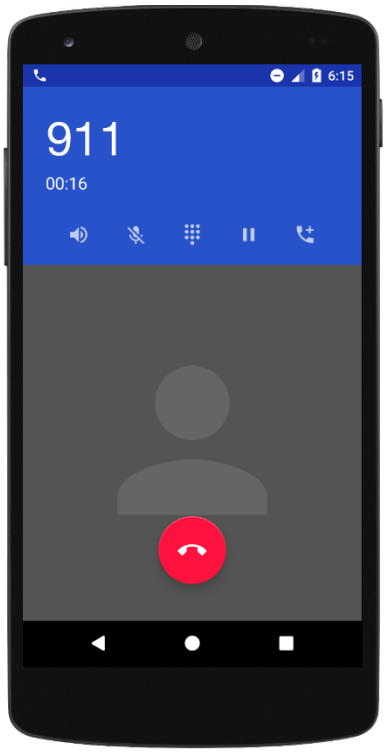

Figure 5.7. On Call

References

[1] "Autism spectrum disorder: Communication problems in children," in NIDCD, NIH, 2016. [Online]. Available: https://www.nidcd.nih.gov/health/autism-spectrum-disorder-communication-problems-children

[2] A. Ramstead, “ASD and First Responders: Bridging the Gap,” Autism Asperger’s Digest, 2012. [Online]. Available: http://autismdigest.com/asd-and-first-responders/