Final Report
Social Interactome Breathalyzer
“Breathe-EZ”

CS 4624 Spring 2016 – Dr. Edward A. Fox
Virginia Tech – Blacksburg, VA 24061
April 25, 2016

Client

John Crawford, Addiction Recovery Research Center
Mikhail Koffarnus, Addiction Recovery Research Center

Project Team

Chris Cornett – Graphical User Interface Developer
Ben Robohn – Hardware/Software Interaction Developer
Aska Toda – Camera Interface Developer
Colton Walker – Graphical User Interface Developer

Table of Contents

Table of Figures

Table of Tables

1. Executive Summary

The breathalyzer application is part of an ongoing research project Mikhail Koffarnus is pursuing as a research professor at the Addiction Recovery Research Center (ARRC). Participants, which may number in the hundreds, will participate in an alcohol addiction recovery program, which includes random breathalyzer tests. For each test a participant passes, they will be monetarily rewarded. The amount they are rewarded will increase with each successive passed test. The hope is that continued clean tests incentivized by monetary rewards will aid and motivate users in their road to recovery.

The most important form of data being collected by this application is the user's blood alcohol content. While the application is primarily responsible for collection, the data will be processed in several ways after collection. The ARRC will be responsible for database interfacing and processing. They will check that the user has a blood alcohol content of 0.00 and use images to identify that the participant is the correct person. If the user does have a blood alcohol content of 0.00 and is correctly identified, they will be paid for the successful measurement.

In order for the ARRC to be able to identify participants, the application must take pictures. The camera is set to take three photos of the participant as they use the BACtrack breathalyzer. By taking these pictures during the measurement process, the application ensures that the BACtrack device will be in the picture with the user. The application will then store the picture that it finds has the highest confidence of face detection as determined by an algorithm. This picture is important because it will help the ARRC confirm the user's identity, keeping users from easily exploiting the system by having a sober friend blow for them.

2. Users' Manual

In order to get the full Breathe-EZ experience, it is recommended that users participate in the program being offered by the Addiction Recovery Research Center. Breathe-EZ works in conjunction with a BACtrack breathalyzer device, which the ARRC will provide. The application will notify the user when it is time for them to measure their blood alcohol content. Users will have twenty minutes from the time of their notification to take a measurement. In order to take a measurement, the user must follow a simple series of tasks:

1. Hit the "Connect" button.
2. The TextView at the bottom will display "Status: Connected".
3. Hit the "Measure" button.
4. The TextView will display a 12 second countdown.
5. At the end of the countdown, the TextView will display "Center face and start blowing." Center the camera on your face and start blowing.
6. Keep blowing with your face in the frame until the TextView displays your BAC.

If the user is participating in the ARRC study, their measurement results will be sent to the ARRC, along with a photograph of the participant using the BACtrack breathalyzer. This will be used to determine if the user will be rewarded for their participation and sobriety.

3. Developer's Manual

GitHub Repository: https://github.com/brobohn/Breathe-EZ

3.1 Inventory of Software and Data:

- Lines of code: 436
- Application size: 364kb

- **java/**:
    - **MainActivity**: The Java code for the application's functionality. An in-depth description of the class follows below.

- **res/**: Contains the XML files for the Android project.
    - **layout/**: contains the layouts for the screens in the Android Application
        - **activity_main.xml**: The main activity screen.

    - **menu/**: contains the XML code for the dropdown menus on different screens.
        - **main.xml**: The XML code for the dropdown menu on the main activity

    - **values/**: XML files that neatly organize different variables used in the design of the application in a way that helps to keep values consistent.
        - **colors.xml**: Stores the hexcode RGB values under easily accessible names.
        - **dimens.xml**: Defines sizes, such as margin sizes, as variables.
        - **strings.xml**: Stores all of the strings used in the layout in easily accessible variables.
        - **styles.xml**: Defines the style of different aspects of the design.

Implementation Details for MainActivity.java

The MainActivity Class is loaded on startup of any Android application. Our particular app incorporates several components into the MainActivity class: the Camera Module, the Android FaceDetector, and the BACtrack API.

The Camera module used is android.hardware.camera. Google has introduces a new Camera module called android.hardware.camera2, which starting in Android 5.0 Lollipop replaced the older module. This new module does not work on pre-Lollipop devices, which comprise 59.6% of Android devices currently in use. Therefore, the older Camera module was used for this app, which for now functions perfectly on the Android 6.0 device used for testing. In the future, the old camera module will presumably stop working, and the new one will have to be implemented.

The FaceDetector module used is android.media.FaceDetector. This module does not support facial recognition, only detecting the presence of a face in a given picture.

The BACtrack API was provided by BACtrack at no charge. It provides a set of methods for communicating with the Bluetooth-enabled BACtrack device.

The flow of the program is as follows: when the app launches, it checks for a Bluetooth connection. If Bluetooth is turned on, the user may tap the “Connect” button. This calls a BACtrack API function to connect to the nearest BACtrack device. When it is connected, the user may tap the “Measure button”. When this button is clicked the camera is started. A preview of the camera is displayed on the screen, and the user is prompted to center their face in the preview. Since this preview is embedded in the MainActivity, only a low-res thumbnail version is displayed on the screen. On this same click, a countdown of 12 second (recommended by the BACtrack API) is started to give the BACtrack device time to warm up. At the end of this countdown, the user is prompted to blow into the device to take a BAC measurement. Also at the end of this countdown, the camera will start the process of taking three pictures at 1.5 second intervals. When all three of these pictures are taken, the FaceDetector will pick the picture with the highest face detection confidence. The measure BAC level will be displayed, and a Toast will be presented saying that a picture has either been saved to the gallery or rejected because no face was detected.

3.2 Use Cases

Note: the blue numbers in the bottom left corner of the boxes represent the figure numbers for their corresponding screen dumps.

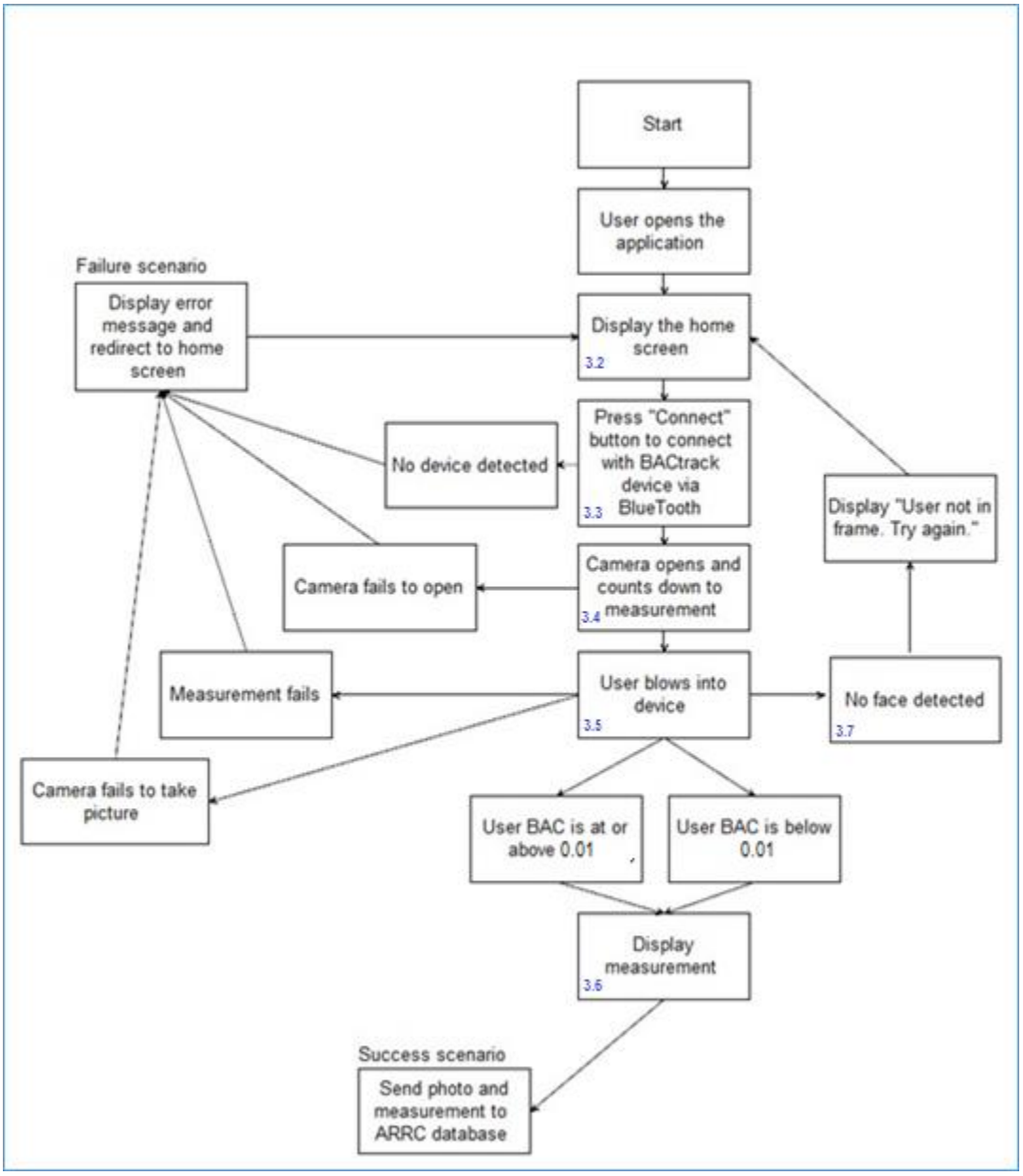

Figure 3.1: Flowchart detailing use cases

3.3 Hardware Requirements

This application requires two pieces of hardware.

Smartphone: A smartphone running Android 4.0+ is required to run the app. It also needs a front-facing camera and Bluetooth capability. Most development and testing is being done on a Motorola Nexus 6. Samsung devices are not supported.

BACtrack Breathalyzer: The app is compatible with any BACtrack breathalyzer. Development and testing are being done with the BACtrack Mobile Pro.

3.4 Software Requirements

This application will take advantage of several free and open-source tools.

Android Studio 1.5.1: Application development will be done in Android Studio v1.5.1. This IDE is available for Windows, Mac, and Linux.

Android SDK: This application will be developed using a minimum of version 18 of the Android SDK with a target of version 20.

BACtrack SDK and API: BACtrack makes its SDK and API available to parties requesting permission to use it. It allows a mobile application to connect to a BACtrack breathalyzer.

3.5 Screen Dumps

i. Open Breathe-EZ:

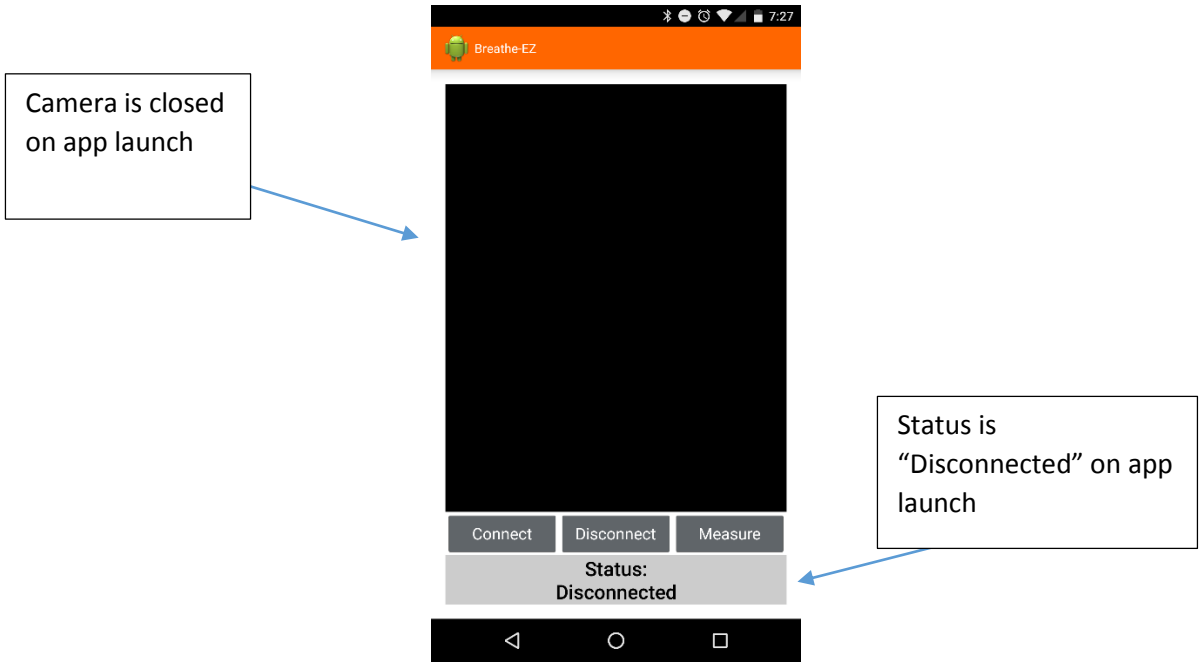

Figure 3.2: Breathe-EZ Screen 1

ii. Press "Connect" to connect to BACtrack device:

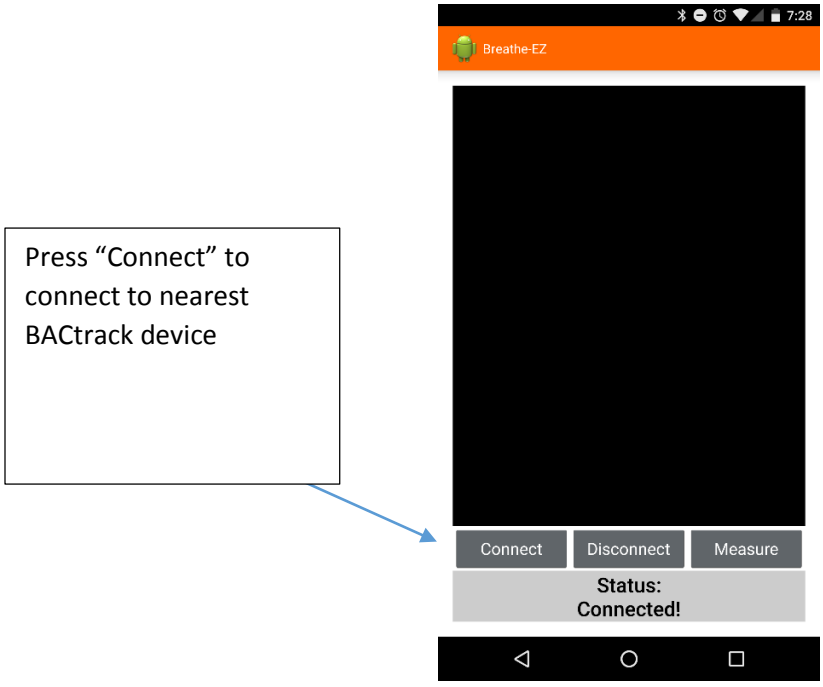

Figure 3.3: Breathe-EZ Screen 2

iii. Press “Measure”:

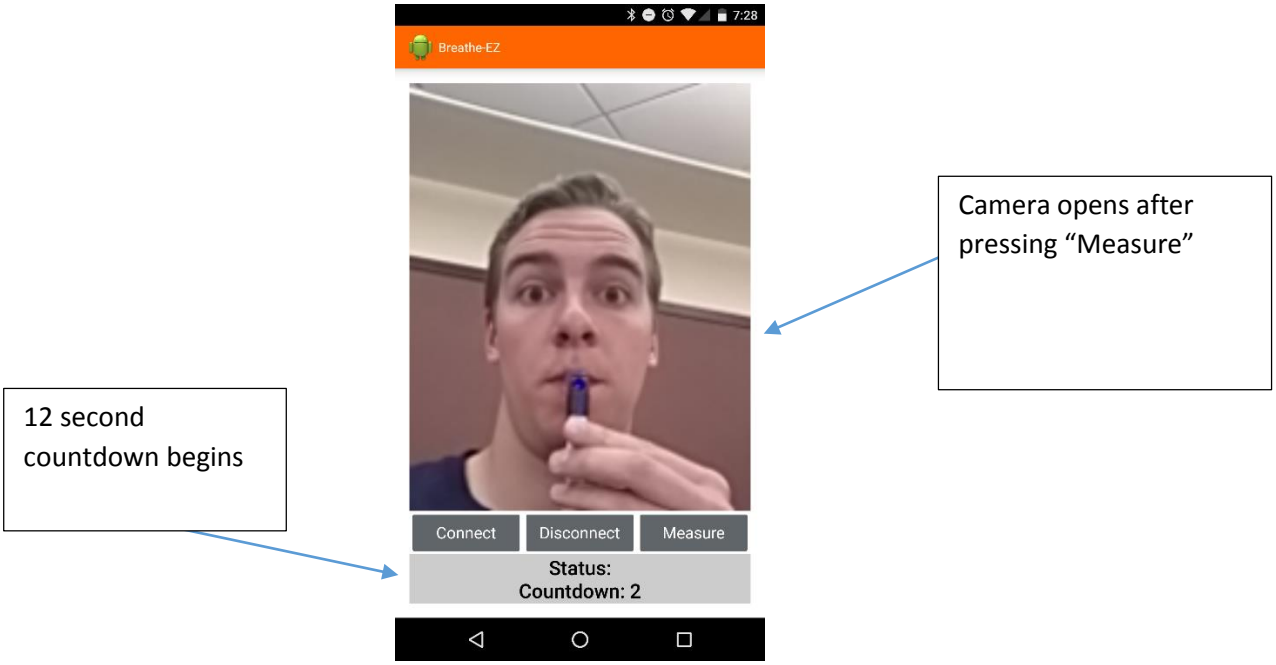

Figure 3.4: Breathe-EZ Screen 3

iv. After countdown, start blowing and keep your face in the camera preview:

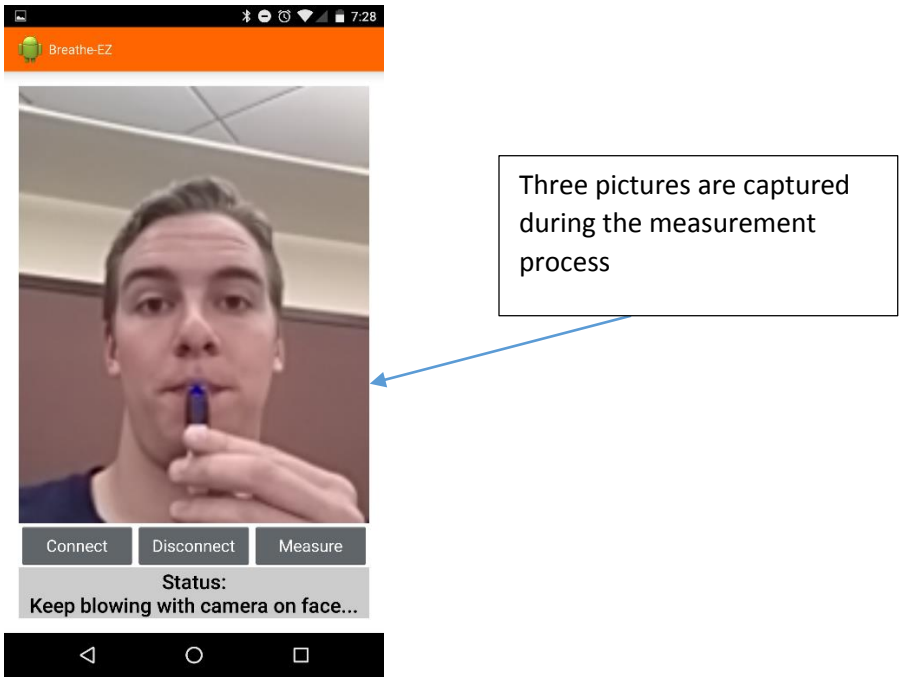

Figure 3.5: Breathe-EZ Screen 4

v. Displays BAC and accepts or rejects picture:

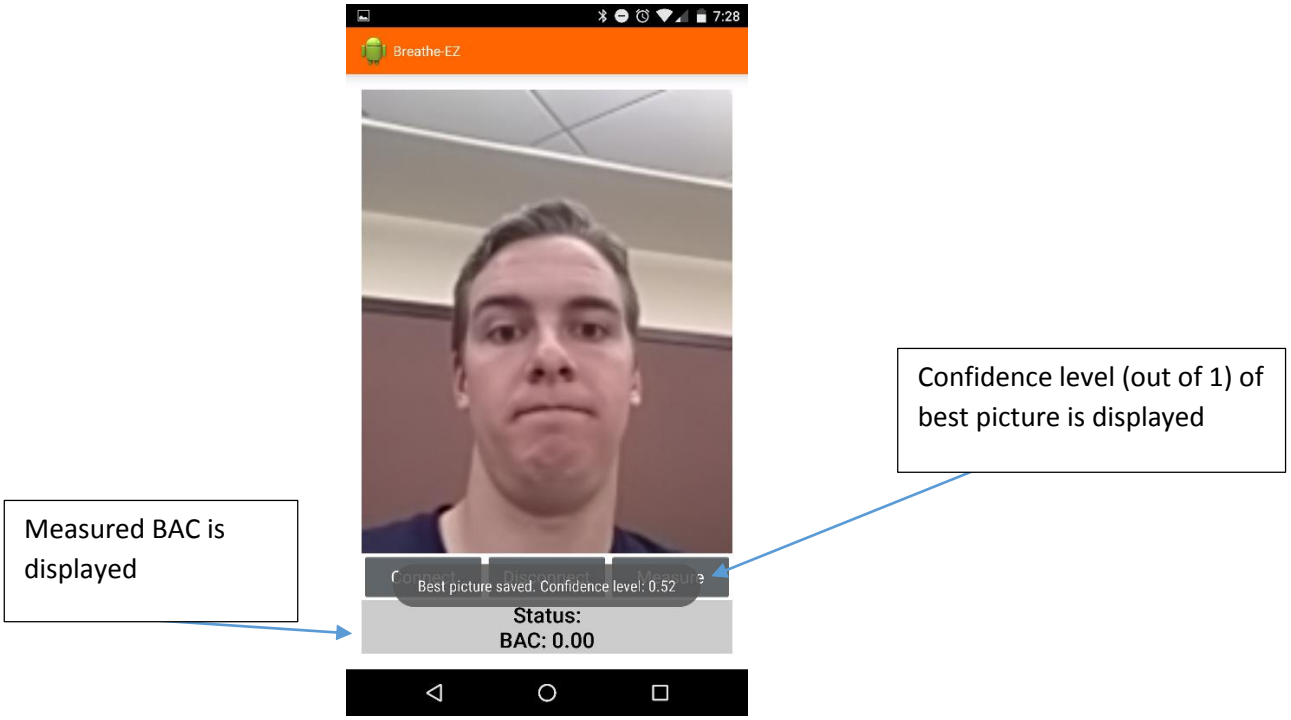

Figure 3.6: Breathe-EZ Screen 5

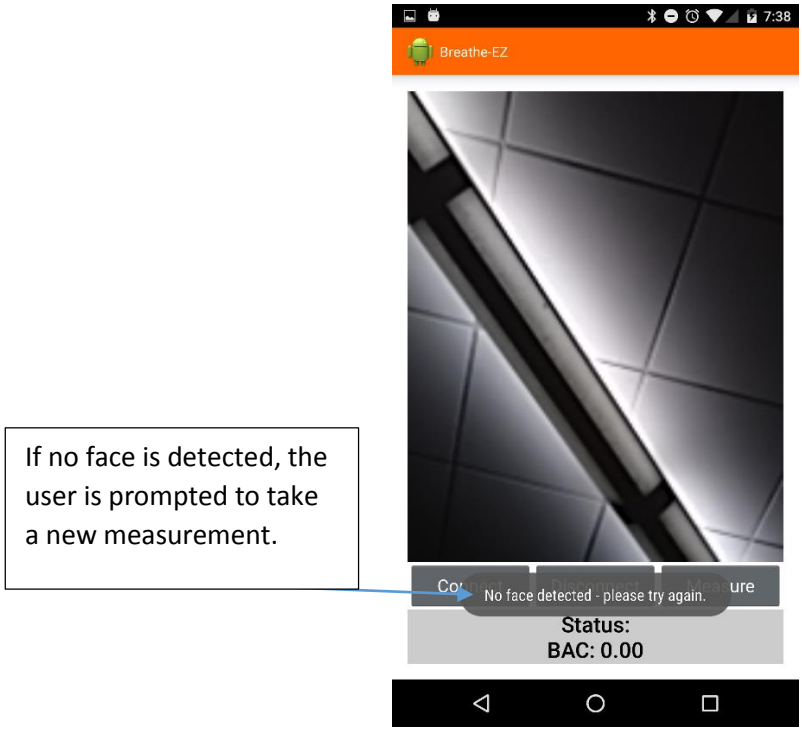

Figure 3.7: Breathe-EZ Screen 6

3.6 Future Work for the ARRC:

- Implement a notification system to let users know when they're required to take measurements.
- Implement database interfacing and processing in order to collect the data that the application currently stores locally.

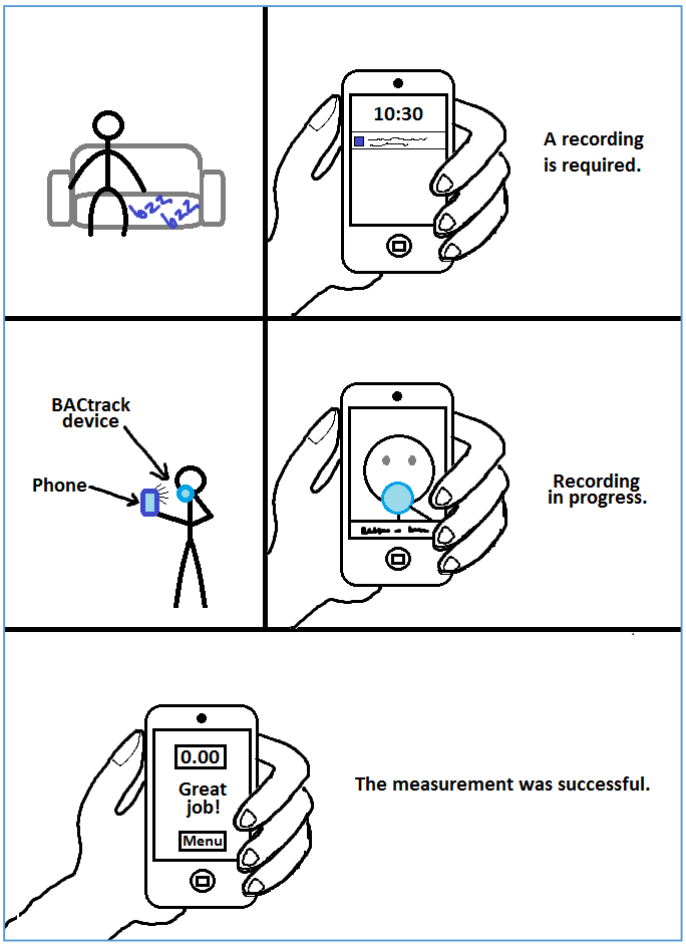

Figure 3.8: Storyboard illustrating notification system.

4. Lessons Learned

4.1 Requirements

Requirement	Priority	Completed?
Build application for Android.	High	Yes
Include documentation for building the app in Android Studio.	High	Yes
The app must interface with the provided BACtrack device.	High	Yes
The app must include a “Measure” button that guides users through the measurement procedure.	High	Yes
The app must photograph the user during their measurement	High	Yes
The app must use facial detection to be sure there’s a visible face in the picture	High	Yes
It would be nice if the application can sense the blue LED of the BACtrack device. This feature is not expected.	Low	No

Figure 4.1: Requirement Table

4.2 Implementation Schedule

Tasks	Date of Tasks (by Weeks)											
Task 1: Implement a basic graphical user interface for the application.												
Task 2: Implement the ability to connect to a BACtrack breathalyzer device via Bluetooth												
Task 3: Implement the ability to take a measurement using a BACtrack breathalyzer device												
Task 4: Implement the ability to open an Android phone's front-facing camera												
Task 5: Implement the ability to keep the front-facing camera open while taking a measurement												
Task 6: Implement facial detection during measurement												
	08	15	22	29	07	14	21	28	04	11	18	25
	Feb				Mar				Apr			

Figure 4.1: Gantt Chart

4.3 Problems and Solutions

A. **Problem:** When we began developing Breathe-EZ, we thought we had a good idea of how the finished project would work. However, throughout development, we were frequently presented with issues that we hadn't considered before. Among these were questions about how the home screen should be structured, what the maximum acceptable BAC for a user would be, and which metrics to use for determining whether a picture of a face is acceptable. There were several times that we were forced to halt program development until a clear solution to these issues was decided.

   Solution: We started scheduling our meetings with John Crawford more than a few days in advance. At each meeting, we would decide on the next meeting time. At these meetings, John was very good about delivering definitive answers to our questions. He was also very responsive over email, which helped us continue development on more than one occasion.

B. **Problem:** During our early prototypes for Breathe-EZ, the application could open a camera preview from the front facing camera by launching a new screen. When we showed this to John, he instructed us to embed the camera directly into the home screen. He wanted the camera preview to open when the measurement process was initiated without having to launch a new screen. The Camera API doesn't provide a simple way to do this.

   Solution: After researching the topic, we found that it is possible to embed the camera preview within a screen using a custom SurfaceView. By doing this, we implemented the ability to display the camera preview from the home screen.

C. **Problem:** Although implementing our camera using a SurfaceView allowed us to display the video feed, this presented a new problem. The display was rotated 90 degrees clockwise. Furthermore, the images were being rotated 90 degrees counterclockwise.

   Solution: It took us a while to figure out what the problem was here. It still seems strange that the video feed isn't displayed upright by default. Eventually, we figured out that we could set the display orientation and the actual camera rotation separately. Strangely, using a Samsung phone required us to rotate each of these properties by an additional 180 degrees, but we had bigger issues with Samsung than that.

D. **Problem:** Although our group didn't have much experience with Samsung phones, it is well known to Android developers that Samsung's devices perform differently. The main issue is that the android.media.FaceDetector library is not compatible with Samsung devices. This library was a big part of our design strategy, so this was a big issue.

   Solution: The ARRC has agreed not to supply participants with Samsung devices. Because facial detection was such a big part of our assignment, they didn't think we should have to implement facial detection all over again just to work with one specific manufacturer.

E. **Problem:** In the early stages of this project, we were hoping to implement facial recognition by comparing the pictures taken during measurement against an existing image of the intended user. Facial recognition turned out to be quite a complicated task, and the majority of libraries available for use were too expensive to use.

   Solution: John Crawford told us from the beginning that facial recognition may be an unrealistic goal. Together, we modified the program requirements so that the application would perform facial detection but not facial recognition. We used facial detection to determine the number of faces in a given image. This is used to require that there is only one face in the frame during the time of measurement. The ARRC members in charge of determining monetary rewards will manually check whether the pictures sent with measurement data contain the correct user and the blue LED of the BACtrack device.

5. Acknowledgements

Edward Fox, Professor
Virginia Tech
Email Address: fox@vt.edu

John T. Crawford, Research Programmer
Virginia Tech Carilion Research Institute – Addiction Recovery Research Center
Email Address: jcrwfrd@vtc.vt.edu

Mikhail Koffarnus, Research Assistant Professor
Virginia Tech Carilion Research Institute – Addiction Recovery Research Center
Email Address: mickyk@vtc.vt.edu

National Institutes of Health, Sponsor
Grant Title: Remote Alcohol Monitoring to Facilitate Abstinence Reinforcement
Project Number: 5R21AA022727-02
URL: https://projectreporter.nih.gov/project_info_description.cfm?aid=8904569

6. References

BACtrack. "BACtrack Mobile Pro." *BACtrack Mobile Breathalyzer for IPhone & Android Devices*. N.p., 2016. Web. 1 Feb. 2016. <https://www.bactrack.com/products/bactrack-mobile-smartphone-breathalyzer>.

BACtrack. "Build Apps with BACtrack." *BACtrack*. N.p., 2016. Web. 1 Feb. 2016. <https://developer.bactrack.com/>.

Google. "FaceDetector.Face." *Android Developers*. N.p., 2016. Web. 18 Apr. 2016. <http://developer.android.com/reference/android/media/FaceDetector.Face.html>.