HealthMap

Niharika B, Karen T, Phoebe F, Manikanta K

Stay confident and accountable about your health before getting behind the wheel

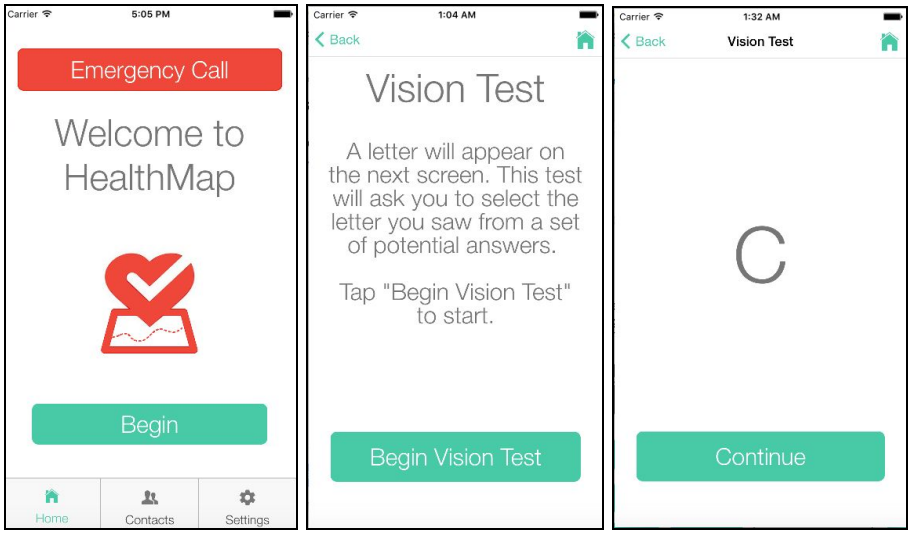
Problem and Solution Overview

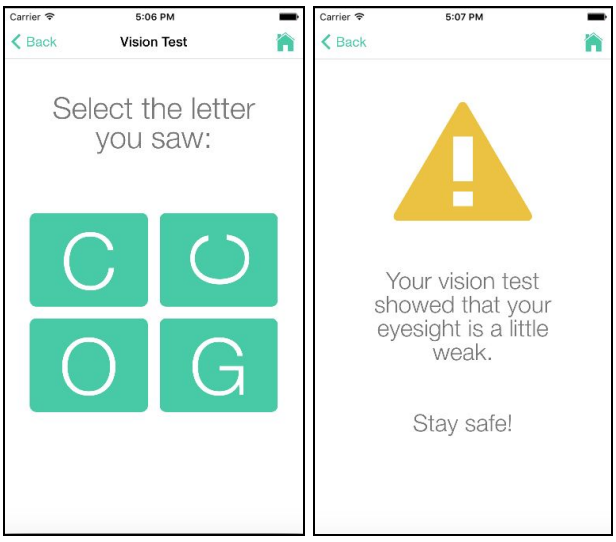
At one point or another, everyone has been a bit too tired, ill, or under the influence to drive. Though it may seem easier to just get in the car and drive, those actions can have negative consequences for you and your fellow drivers. HealthMap is an intuitive and easy health verifier that tests your motor skills and helps you make a more informed decision before hitting the road.

Tasks and Final Interface Scenarios

1. I want to know how to be a safer driver (simple)

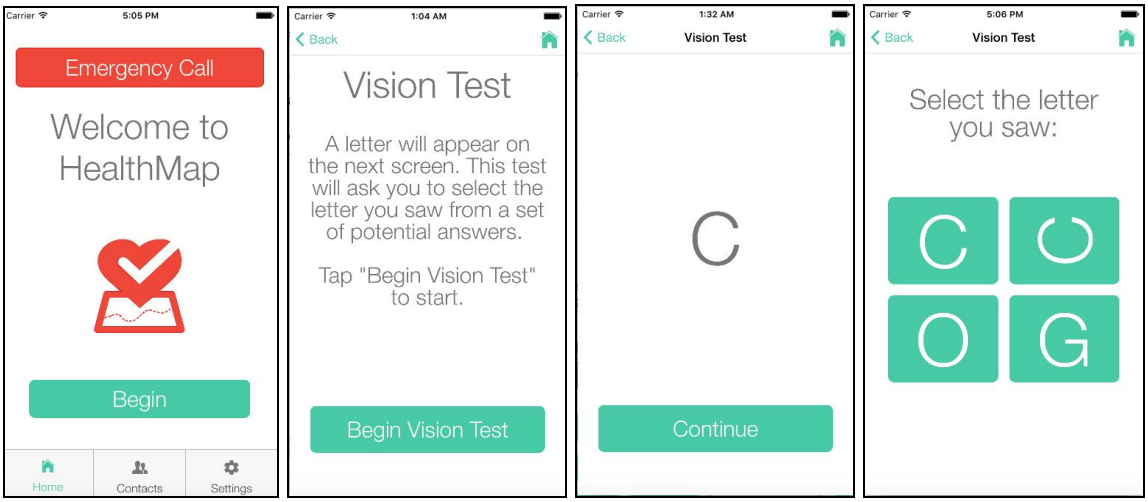
We wanted to create a task that would get users to use the app on a consistent basis. No matter how well rested and energized you feel, we believe that erring on the side of caution and checking your health is important. So, we wanted to find a way to help users who had room for improvement.

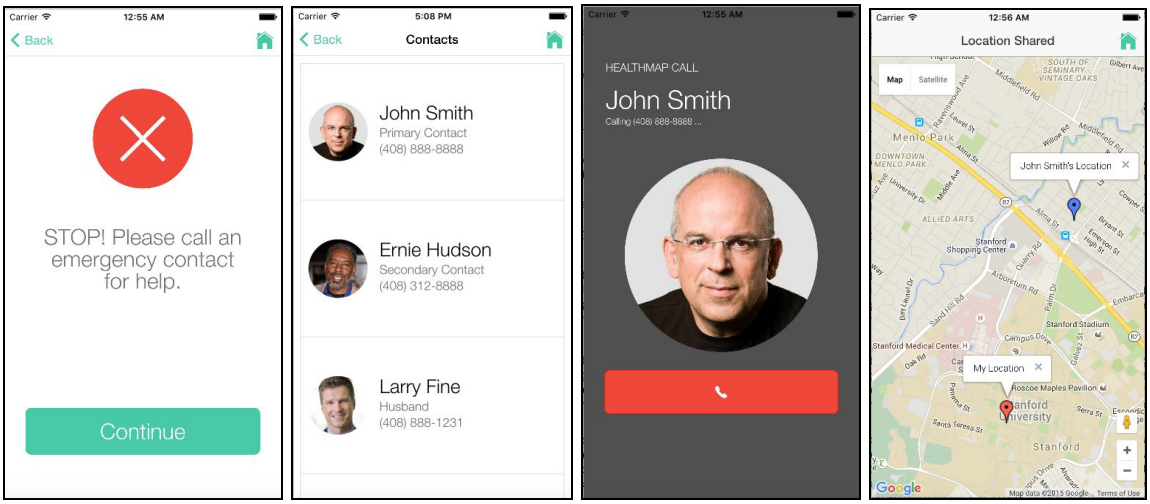




2. I want to get help when I need help (medium)

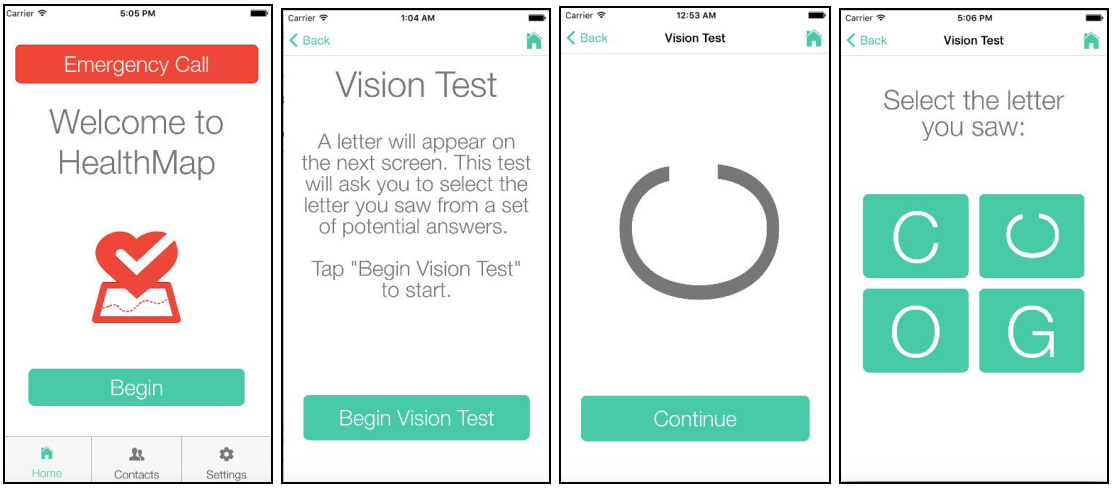
We wanted to create a task that would allow users to immediately get help if necessary. Instead of getting the results and then having to leave the app and call an emergency contact normally, we wanted to allow users to do everything in one streamlined workflow. This task allows users to call emergency services or close friends and family if the situation requires it.

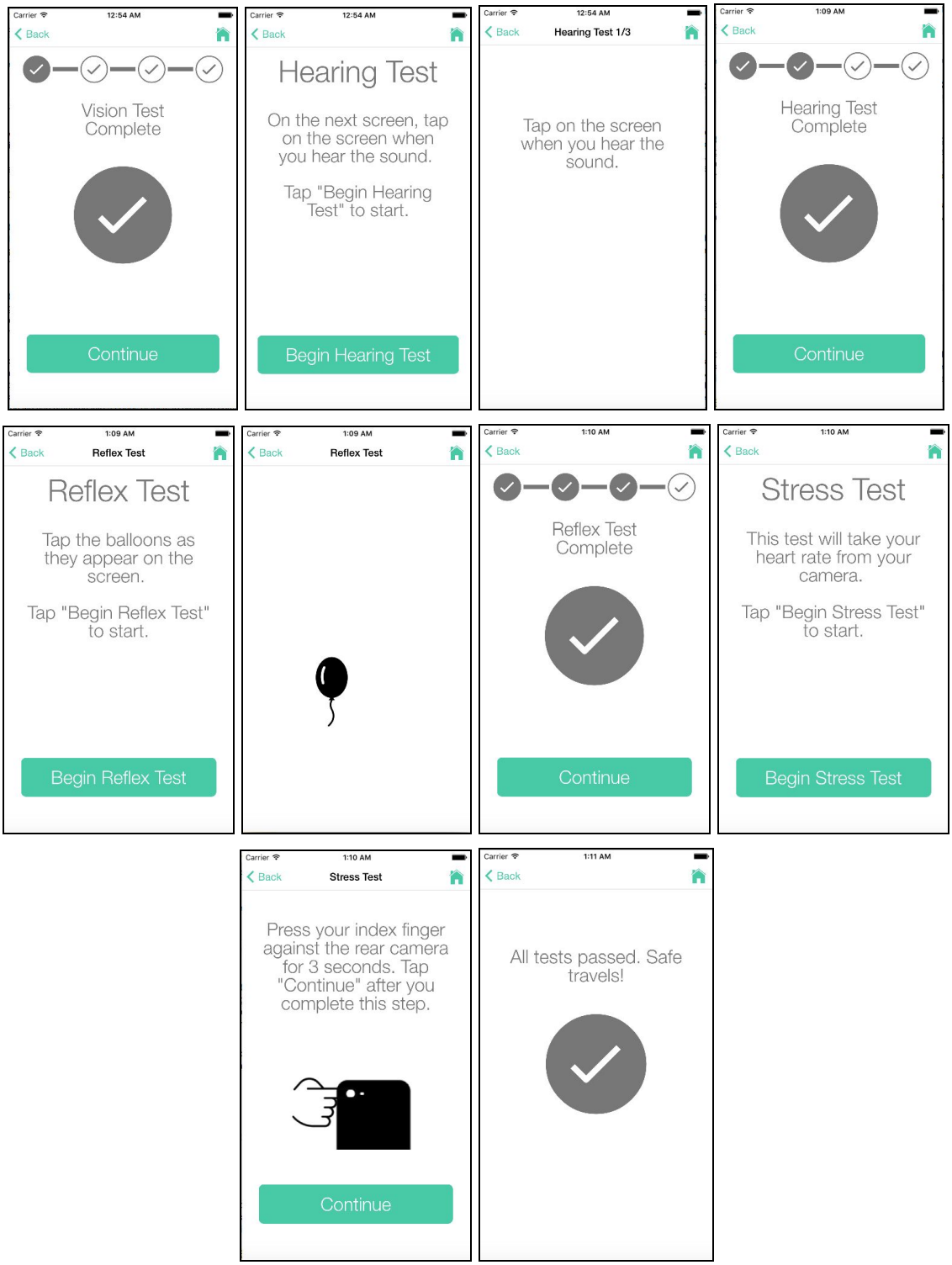




3. I want to know if I am fit enough to drive (complex)

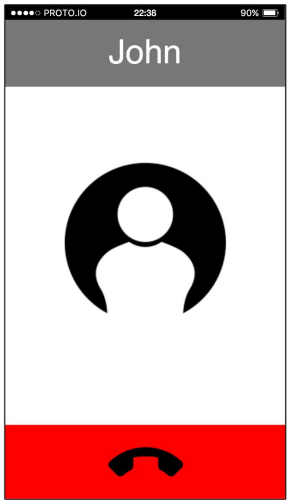
The major task we wanted to implement was the battery of tests that evaluate a user's motor skills. These tests include vision, hearing, reflex, and stress tests. We picked the tests and designed them after doing significant research on industry wide safety standards as well as scientific research on what is necessary to be a safe driver. So, we wanted to combine all the information we had gathered into a series of tests to evaluate someone's ability to drive.

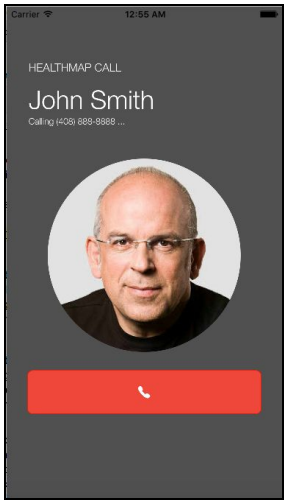


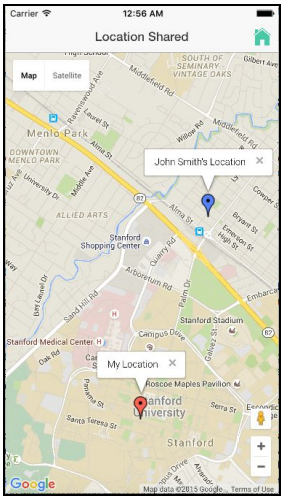


Design Evolution

Lo-fi prototype -

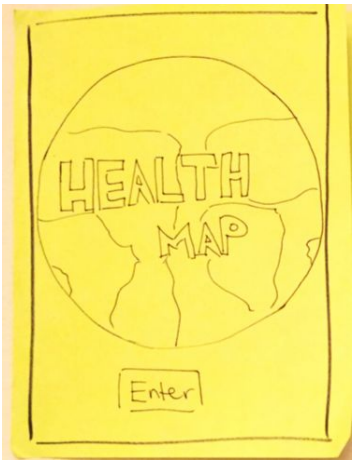


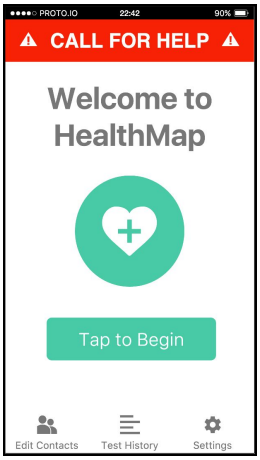


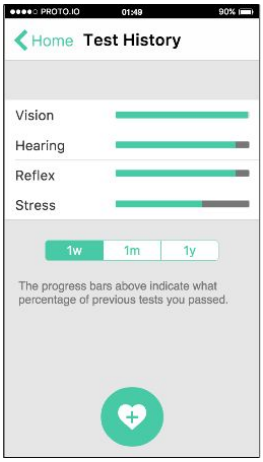


We did user testing on our low fidelity prototype, and found lots of interesting insights about user behavior and preferences. We found that our users were very unwilling to read blocks of text, and rather skimmed over instructions - they would only go back to them if they found themselves unable to complete the corresponding task. So, we were very conscious of creating short, clear text instructions throughout the application.

Secondly, we got some extremely useful input about what a user would like out of a health verifier. Specifically, the ability to bypass the tests and go straight to an emergency call as well as the ability to store health history. These were both elements we had missed when originally designing the application, but we all agreed that we thought these were extremely important additions.



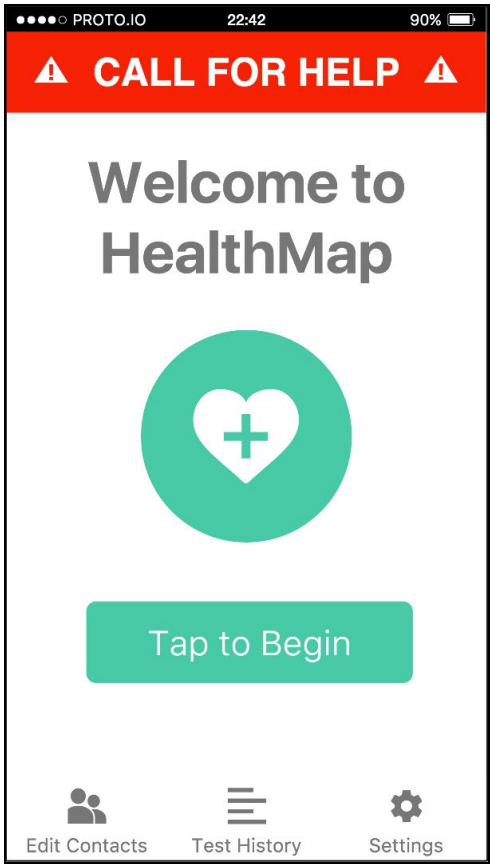




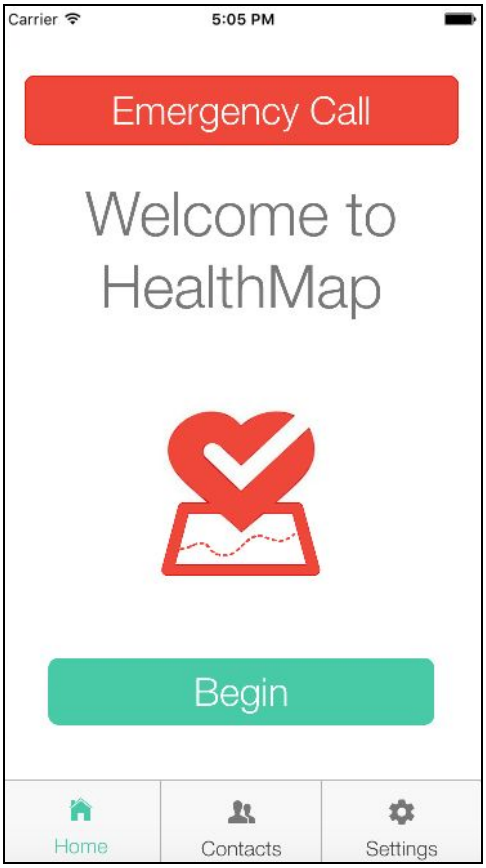
Major Usability Problems Addressed

Icon Confusion - Severity 4

Multiple users reported that the green color of our icon was confusing, and made it seem like the icon was a button. So, we changed our icon to be a different color and shape. The rationale behind changing the shape was to prevent users from believing it was a button, and the color change to red was to further differentiate it from the green buttons present in our application.



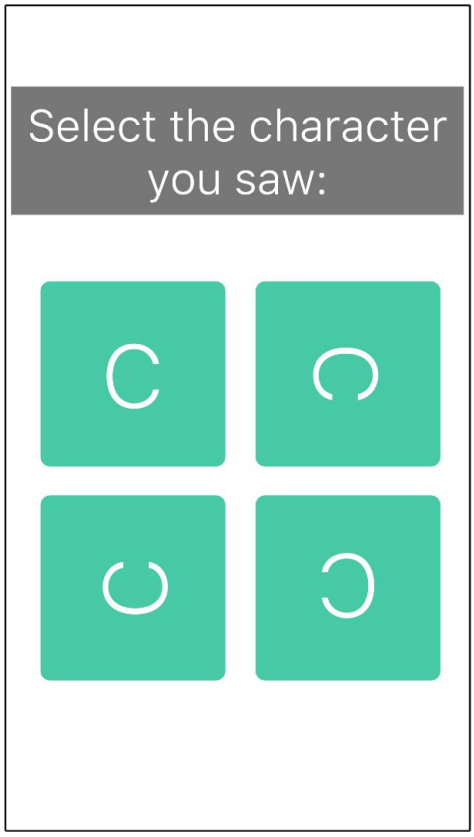
Original screen with misleading icon



New screen with less ambiguous icon.

Lack of Exit Screen - Severity 3

Our original interface did not allow a way to exit the current evaluation once the test was started. We agreed that this was an oversight, as users may get distracted or occupied during the test and have to restart from the beginning. Our original implementation did not allow for this option. To change this, we added “Home” buttons on every screen of the test.

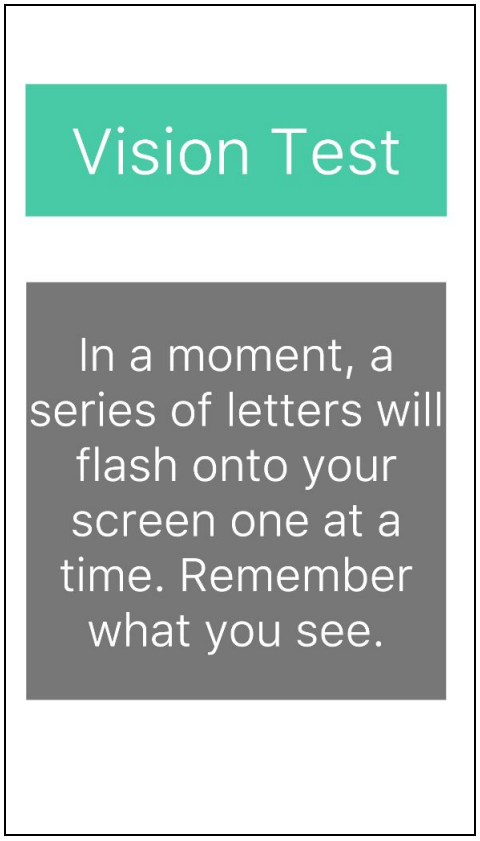


The original screen - no way to exit

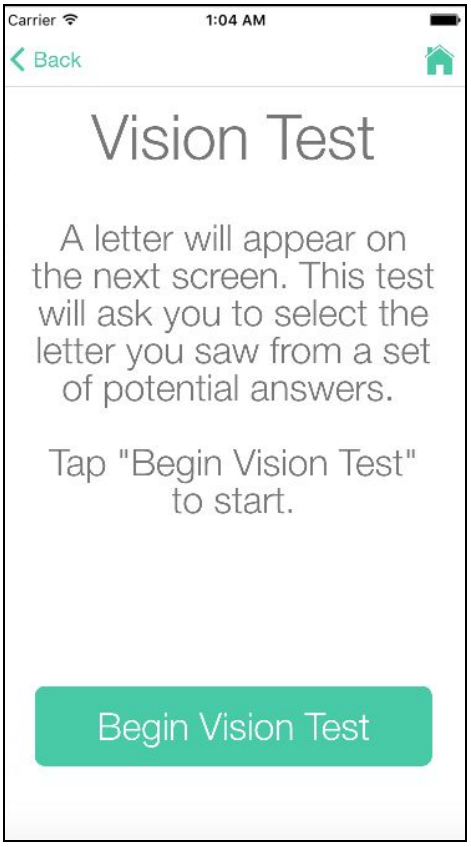
The screen with the added home and back buttons.

No “Continue” Button - Severity 3

In the original application, the screen progression for each task in the test was set to a timer. Users were unable to replay the tests if they were momentarily distracted or occupied. So, we decided to change the timer aspect to a “continue” button that would allow a user to manually proceed through the test.



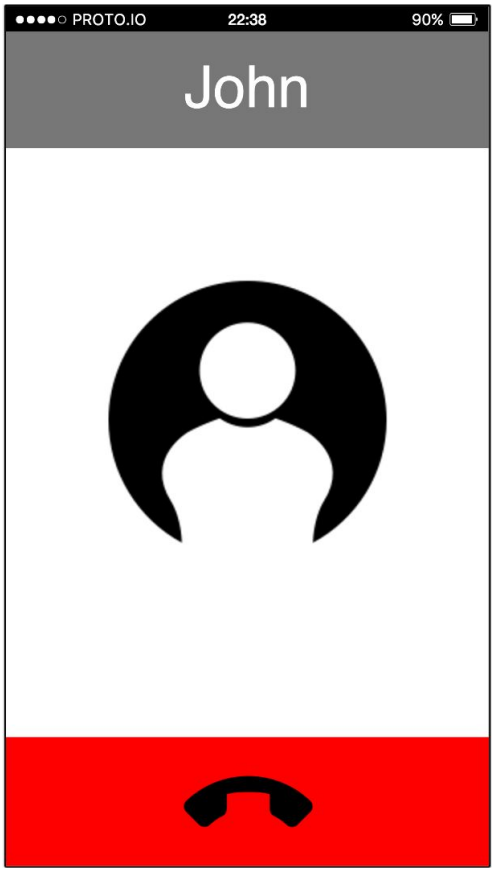
Original screen without continue.



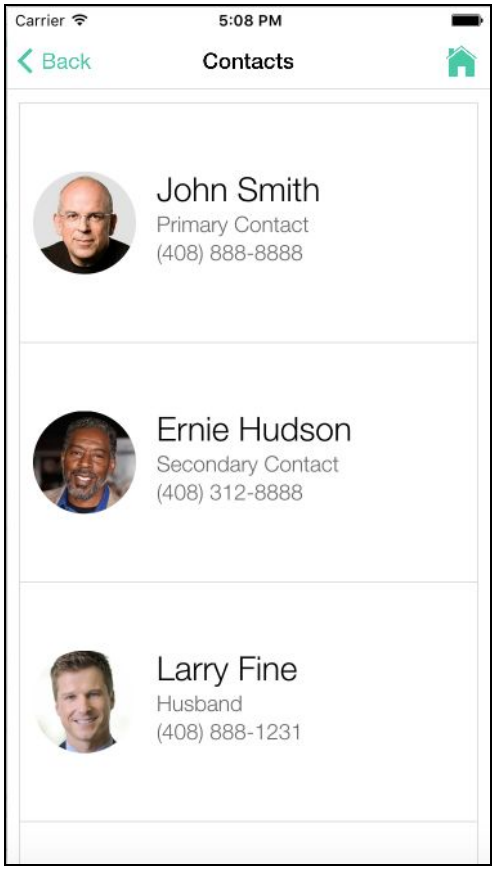
Continue button allows for user control over progress.

Cannot Choose Emergency Contact - Severity 3

Our original application made the decision of who to call without asking the user for his or her input. Users were wary about this feature, as you may want to call different people based on the situation - for example, a parent would likely not be preferable if the user is underage and under the influence. So, we decided to allow the user to choose which emergency contact they would like to contact.



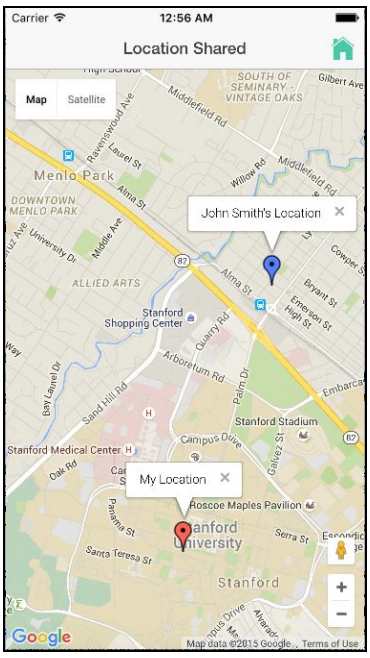
Originally just called first emergency contact.



Now, will open up list of emergency contacts.

Tracking Friend's Location - Severity 2

Users also suggested that tracking the emergency contact's location would be helpful when in this situation. So, we added the ability to track the emergency contact's location and see his or her ETA.



No Information about In-test Progress - Severity 2

There was no way for users to determine their progress while in the middle of a test - this was an issue for some users, as it put the users in a very unclear position. So, we added positioning clues inside all the tests. For example, the first time a user taps after hearing a sound, the hearing test updates from ⅓ to ⅔.

Skipping Instructions for Experienced Users - Severity 2

There was no way for experienced users to skip the instructions in our original implementation, but the addition of the “Continue” button fixed that. Now, users who know what is happening can quickly skip to the test itself.

Prototype Implementation

To implement our prototype, we used Ionic creator, a tool that allows users to create iOS and Android apps by only implementing the HTML, CSS, and Angular.js features. The tool is extremely helpful - they had a simple drag and drop interface that was extremely useful to set up the initial layout of a page. In addition, the tool is excellent at staying up to par with current iOS standards, which makes creating an app much easier. There were a wealth of icons, buttons, and elements to choose from, which made setting up the application’s layout quite easy.

However, the tools were quite finicky to use, and required some hacking to load dynamic content. Most of the content on the app is meant to be static, so trying to dynamically load

names, pictures, and information proved to be a bit tricky. In addition, the emulators created a multitude of problems for us. There are multiple emulation tools available - one that uses XCode's iOS simulator, one that creates a website version of the application, and one that allows you to download and test the app on an iPhone. The XCode simulator and website version were extremely buggy, and often inexplicably did not load. After a bit of finagling, we were usually able to figure out what was wrong, but the dearth of documentation about the website and common bugs definitely hurt us.

We had two major Wizard of Oz techniques throughout the app - the stress test and the emergency call feature. We found an extremely interesting and advanced stress test that uses an iPhone's flashlight and camera to examine a finger and determine heart rate through image processing. However, the code for the tool is not open source, and we would not be able to implement a similar diagnostic tool. So, we decided to Wizard of Oz this technique.

Regarding the emergency call feature, this had to be implemented because it is impossible to make an in-app iPhone call. So, we created a HealthMap call in a similar vein to Whatsapp's "Whatsapp Call." In our app, the call doesn't actually do any real calling, so we had to Wizard of Oz this part to ensure that emergency calling could remain a crucial part of the application.

Currently, the only element that is hard coded is the Address Book Contacts and the contact's location. With the Ionic creator, since we do not have a backend database or server, there is no way to store dynamically generated content. So, we decided to populate the address book with some emergency contacts.

With more time in the future, we would like to expand the customizability of the application. Currently, drivers would have to take the vision, hearing, reflex, and stress test each and every time, which could become repetitive over time. By fleshing out our settings page to allow users to edit the tests they take, we would enhance individual experiences.

Summary

When we originally set out to create HealthMap, we were invested on working on a project that we felt could have actual impact on individuals' lives. So, we set out to research how to create a tool that quickly and intuitively evaluated motor skills. After we first created our low fidelity prototype, it was populated with tests and elements that scientists and researchers had deemed vital for a health verifier. However, doing initial user testing made us realize the importance of diverse perspectives - we got comments regarding additions that we had never even considered before! Slowly, a product that combined expert and regular opinions emerged, and now, 9 weeks later, HealthMap is a product that we feel could and should be used by all drivers. Getting to work on something with true potential impact was extremely fulfilling over this last quarter, and we are now excited to explore how HealthMap does beyond the confines of our CS 147 incubator.