Usability Testing Report - 26/09/2013

On the 26th September 2013, we conducted a usability test with an in development version of our Emergency button app. The aim of this was to see what parts of the interface were unclear to users, what elements that users would expect are missing, and whether appropriate feedback was being given to users.

We had three key tasks that we asked participants to complete and some key requirements to satisfy.

Tasks for each Scenario:
A user should be able to log in and raise an alarm:

- The user should be able to easily identify the correct app
- The user should be able to log in
- The user should be able to find the button to raise an alarm
- The user should know that their alarm has been raised

A user should be able to log in and check in

- The user should be able to easily identify the correct app
- The user should be able to log in
- The user should be able to find the button to check in
- The user should know that their check in has been sent

A user should be able to take control of an unassigned alarm.

- The user should be able to find unassigned alarms
- The user should be able to view the unassigned alarm
- The user should be able to mark an unassigned alarm as dealt with
- The user should be given feedback for dealing with that alarm

Unfortunately, we were not able to gather participants which were within the target demographic - a generally older populace to whom smart phones may not be so natural. We instead used some of our university peers.

Many of the problems identified during the usability test were already known to us, but some new problems also became apparent. It was good to have a second pair of eyes. We have grouped our tests

Instantly we discovered some oversights in our program.
Problem: For the Android app, the hardware back button was not supported.
Priority: High
This is an obvious oversight which we will endeavour to remedy for the next iteration of the

software.

Test 1 - A user should be able to log in and raise an alarm:

Users were expected to login to the app successfully, and then raise an alarm in order to complete the test. Users in general, found it easy to perform the necessary actions to raise an alarm. However, there were some glaring issues.

Problem: The login interface provides no feedback on whether a login has failed to succeed, whether it is due to poor internet connectivity, or a wrong password.
Priority: High
Solution: To remedy this, we will add return messages based on whether the request succeeded or not.

Problem: The alarm interface provides no feedback for if an alarm has sent, or failed to send.
Priority: High
Solution: While the app should take you to the alarm screen if an active alarm has been triggered, there isn't enough feedback whether it was successful if internet connectivity is spotty - we will add a message indicating the phone is attempting to send an alert (eg. a 'sending' message)

Problem: It is too easy to accidentally cancel an active alarm
Priority: Medium
Solution: Implement a confirmation before cancelling the alarm to ensure that this is what the user really wants to do.

Test 2 - *A user should be able to log in and check in*

Users were expected to login to the app successfully, and then 'check in' in order to complete the test. In general, users found it easy to complete the test, but did not receive enough feedback from the program itself on whether they'd successfully completed the task.

Problem: On the main menu, the check-in interface provides no feedback for if the check-in sent, or failed to send.
Priority: Medium
Solution: To rectify this, we will add a notification indicating that the check-in object was successfully received by the server (which was indeed the case during the test, as the expected object appeared in the Parse dashboard)

Test 3 ***-An operator should be able to take control of and resolve an unassigned alarm.***
This test was conducted with a paper prototype. In order to successfully complete this task, participants were expected to login to the operator interface, 'take control' of an active alarm, and successfully resolve it by calling the emergency services and once completed, mark the alarm as resolved.

Participants were generally able to navigate the interface without significant problem, however some suggestions came from our participants.

Problem: The 'dial 000' button should be the smallest, not the largest button (and dial the caree should be the largest button)
This ties into the concept that size should be related to priority or importance. Dispatching the emergency services should be the very last resort -- this is when all else fails.
Priority: Low
Solution: We will proportion the buttons appropriately based on their importance, by making the 'call carer', 'call caree' and 'more details' buttons larger than a last resort button such as 'Dial 000'

Problem: One participant found the 'back' button on the active alert screen 'ambiguous'. Was this relinquishing control of an active alert or just returning to the main menu and still keeping control of the active alert?
Priority: Medium
Solution: In order to separate these two functions and make these differences clear, we will name one button 'relinquish control', and another 'return to main menu'.

Problem: The tab interface on the operator main menu was 'confusing and hard to read at a glance'. Too many tabs.
Priority: Low
Solution: One potential solution identified was to simply colour code alerts based on their status as opposed to having lots of different tabs for the different statuses of alerts. We will take this into consideration as we continue development.

Another suggestion was that instead of tabs on the top (think of an options menu in Windows), we change this to a side tab system in order to navigate between tabs.