GNG2101

Project Deliverable F

Prototype 2

Submitted by

PERSONAL SAFETY. D01, Team 1.5

David Todd, 300194033

Ekene Ebenebe, 8440519

Maya Benhamou, 300173416

Riley de Gans, 300170104

Teo Vogel, 300215037

March 06, 2022.

University of Ottawa

Table of Contents

Introduction

In the last deliverable, we shared our progress on the project so far , and also showcased our first prototype which was mostly a front end effort which guided the development of prototype 2. For this deliverable F, we cover details of our most recent prototype (2), which is an update based on feedback and discovery from our last presentation. With our target specifications in parallel, we contrast the feedback from our most recent client meeting where we did a live demonstration of the Personal Safety App. This document portrays a huge milestone in the progress of our app development as we gear towards the design day. As at the time of this document, we have deployed our application for testing on ios devices, detail of our testing is also covered in this document, as well as next steps towards iterating for the best outcome for the Final prototype for our client.

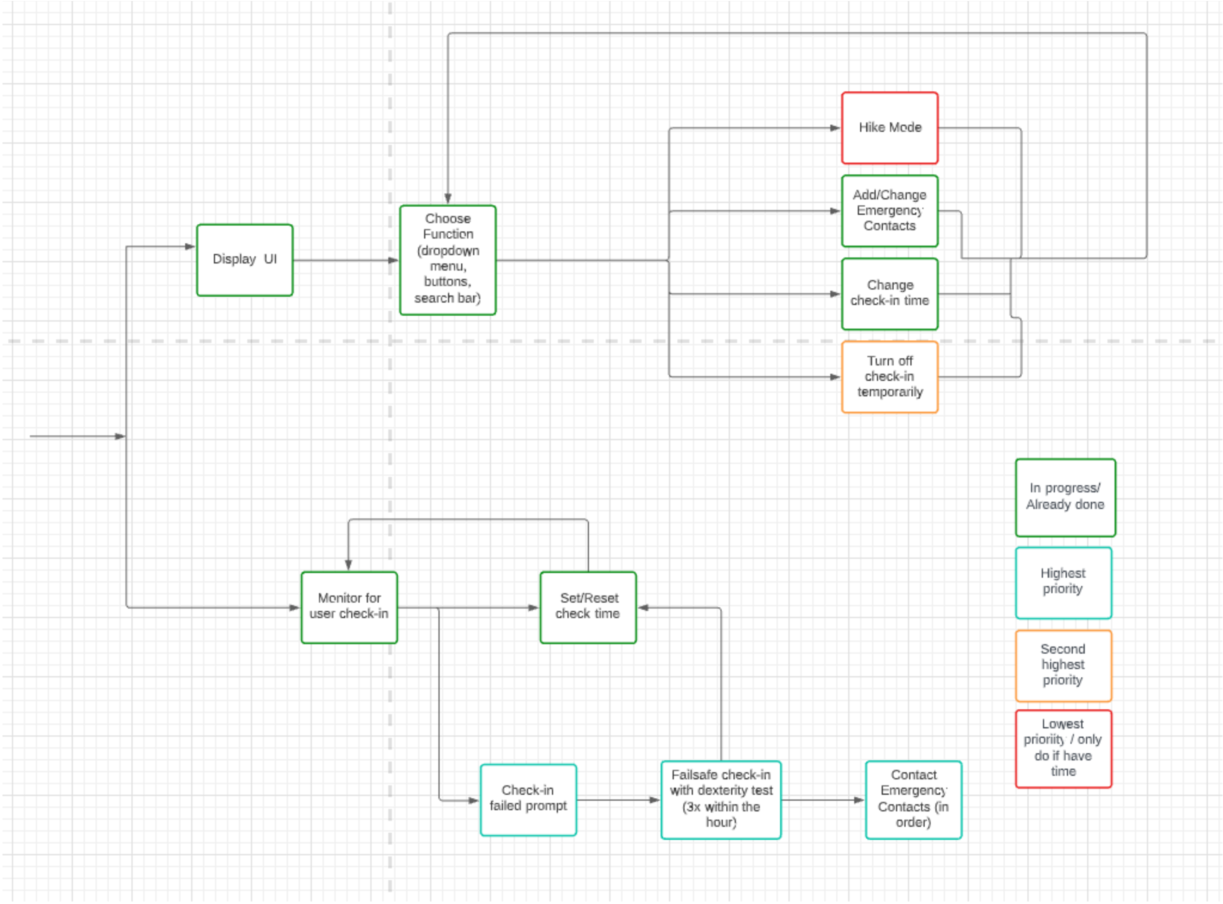
1. Client Feedback

a. Feedback from Client Meet 3

Since the last client meet, we had created a first UI prototype on Canva to help us understand the movement through the app and how it would look like to keep it simple and easy to use as the client wanted. The client was quite happy with the progress that we made so far. We did have questions about some things that our professor had mentioned during our design review class that we wanted the client's feedback on. The feedback from the entire presentation was:

- Priorities: Our team got really lucky by being on the same page about the priorities of things to work on in the next weeks leading up to Design Day and our final product. Currently, the priorities for the entire project are;
    - Functionality → Pause Check-in function → Info bubbles → Hike mode function → Options
- Options: As we are making this product, we realized that some settings have options that can be useful to some users of the application and some can be distracting or useless to others. Some examples of these are :
    - Light vs Dark mode
    - A different colour per page vs continued theme
    - 24-hour clock vs 12-hour clock
    - Contacting one contact at a time vs all of them at the same time
    - Window length change
- Changes: After talking to the teacher after our design review class and our third client meeting we have decided to make certain changes within the application.
    - Instead of using the word failsafe to describe the period of making sure it really is an emergency we have decided to use 5the word “buffer” instead to make it seem a little less intense.
    - The default window of our check-in was originally one hour but after consultation from the client she thought that a bigger window, ie 3 hours, would be better to reach more people and not send out accidental emergency texts to emergency contacts.

b. Updated detail design


Hike Mode
Add/Change Emergency Contacts
Display UI
Choose Function (dropdown menu, buttons, search bar)
Change check-in time
Turn off check-in temporarily
In progress/ Already done
Monitor for user check-in
Set/Reset check time
Highest priority
Second highest priority
Lowest prioritiy / only do if have time
Check-in failed prompt
Failsafe check-in with dexterity test (3x within the hour)
Contact Emergency Contacts (in order)

2. Product assumption

Critical functionality - Assuming that Firebase can consistently check-in on our clients (without paying)

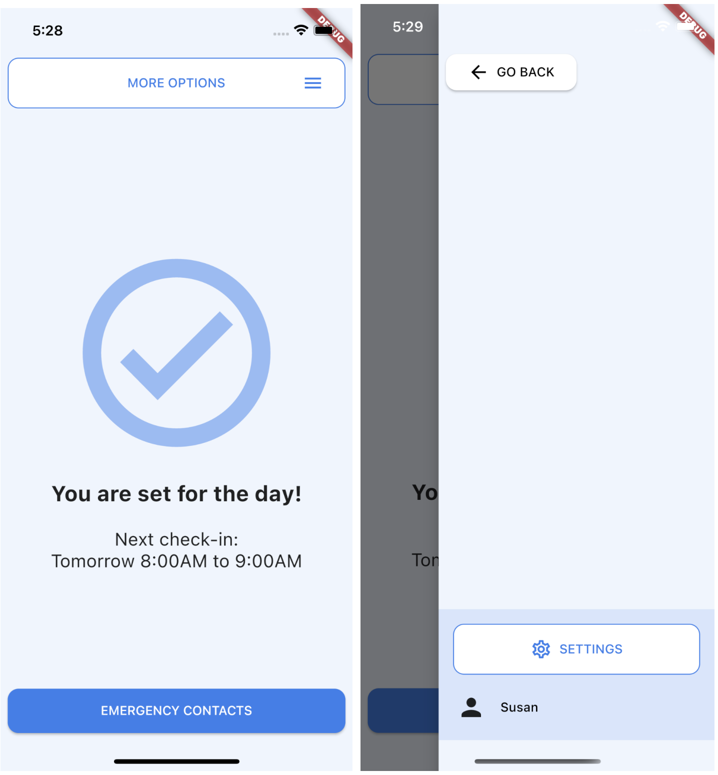
We are assuming that Firebase Cloud Functions has a limitation on the frequency that can be run.

We use this technology to periodically check on the users of the app and detect whether they are safe or not and send an emergency SMS if they need assistance.

Cloud computing time costs money and resources, so it's not feasible to run a task every second to check on the users. So we are planning to throttle the execution of that task, checking on users every 15 minutes.

3. Prototype 2

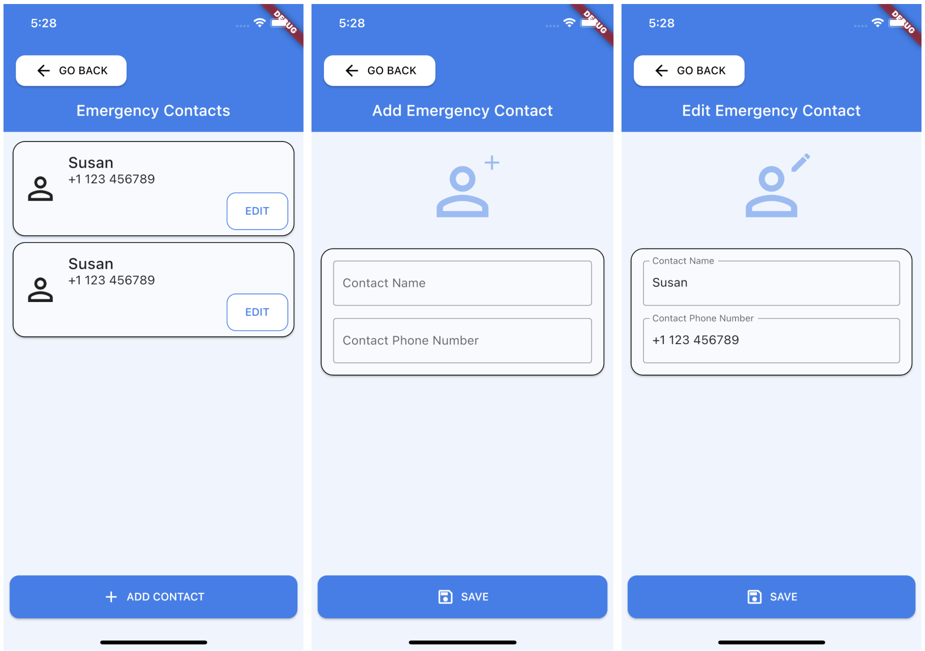
a. User Interface



You are set for the day!

Next check-in:
Tomorrow 8:00AM to 9:00AM

Figure 1 - Home Page programmed in Flutter



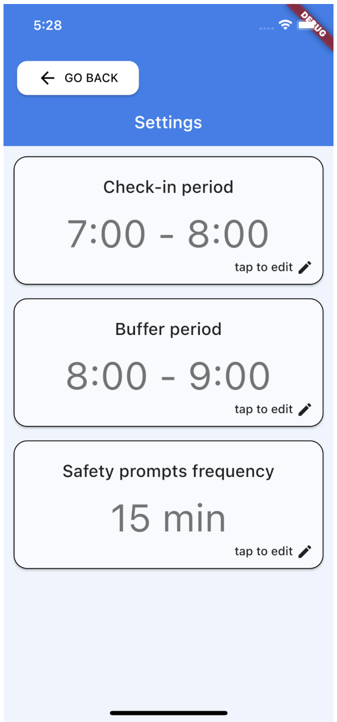
Susan
+1 123 456789

Susan
+1 123 456789

Contact Name: Susan

Contact Phone Number: +1 123 456789

Figure 2 - Emergency contact function and its options

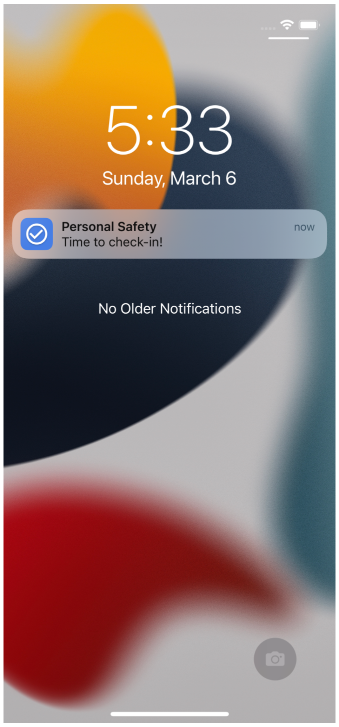


Check-in period
7:00 - 8:00

Buffer period
8:00 - 9:00

Safety prompts frequency
15 min

Figure 3 - Check-in function page



Personal Safety
Time to check-in!

Figure 4 - Check-in notification

b. Prototyping testing

Table 1 : Objectives and results of our prototype testing for our second prototype.

Test Objective	**Target Specification to be satisfied**	**Expected Result**	**Actual Result**
Compare Battery Consumption for Different UI Modes	New Specification from Client - Low Battery Usage	Dark Mode consumes less battery	Both consume close to the same amount, but light mode consumes slightly less
Compare Delay Times for Different Emergency Text Lengths	New Specification from Testing/Research - Maximum Text Length	Longer text messages will have longer delay times	Delay time was independent of text length
Confirm Functionality of Cloud Functions	App communication with emergency contacts and 3 emergency contacts stored	Both true/possible	Both true/possible
Confirm Functionality of Push Notifications	Notification acknowledged, automatic daily reset and user custom check-in time	Both true/possible	Both true/possible

UI Configuration and Battery Testing

For this test the UI from the previous prototype was improved upon in Flutter to create a medium fidelity, general prototype. This prototype was used to deploy the app as an apk to Android users and through TestFlight to IOS users for functionality and usability testing. This prototype was also used to test battery consumption as this was brought up as a concern by the client in the second and third meetings. The test was performed by leaving the app open for specific amounts of time in both the dark (black background, white text) and light (white background, black text) modes, then the battery consumption was recorded at each time interval and plotted below for analysis. Dark mode consumed 0.08% battery per minute while light mode consumed only 0.06%. This shows us that technically light mode is more efficient, but there is no considerable difference, similar to the findings with readability. This means that the choice between these modes will be based on final feedback from the client on aesthetics and usability, as both modes perform very similarly across two different tests.

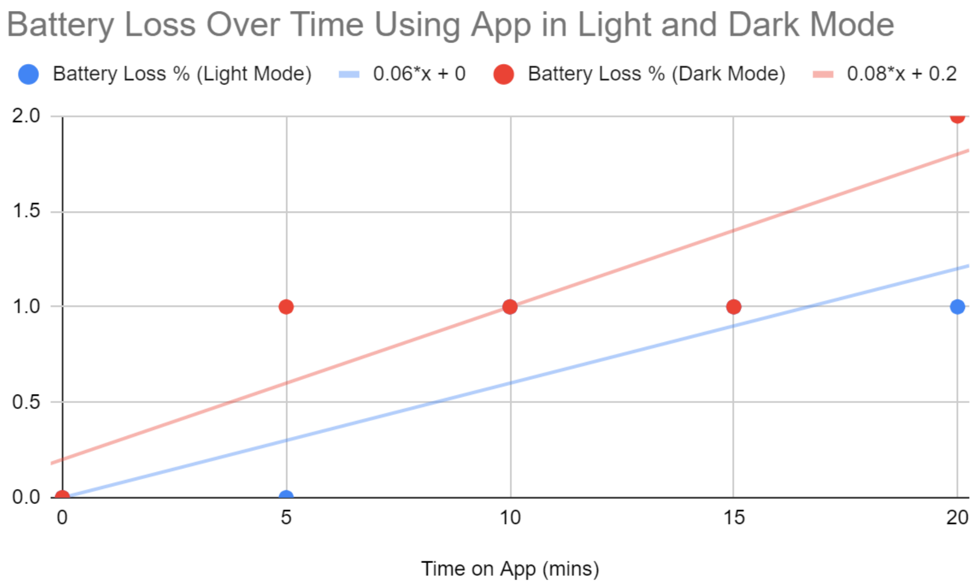
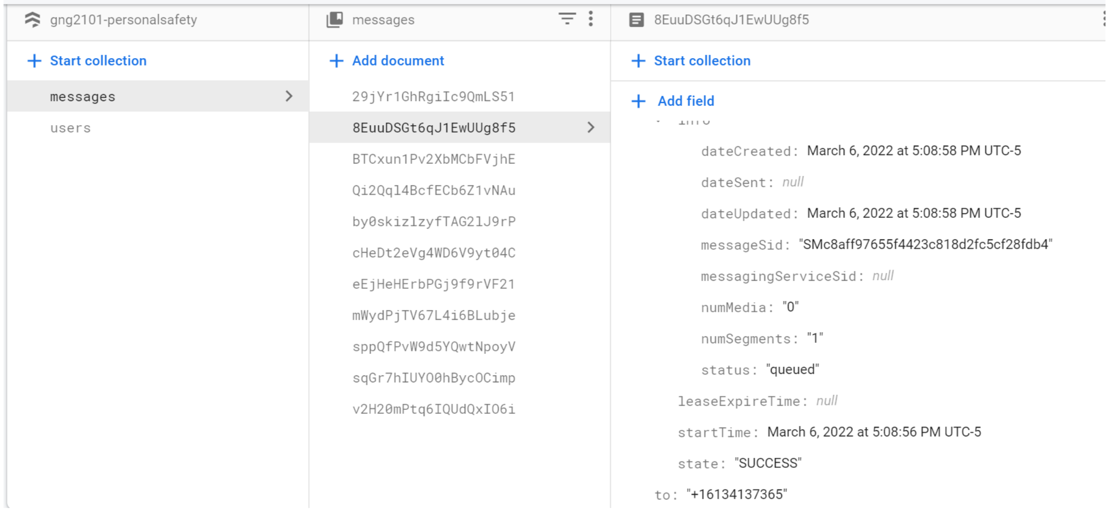


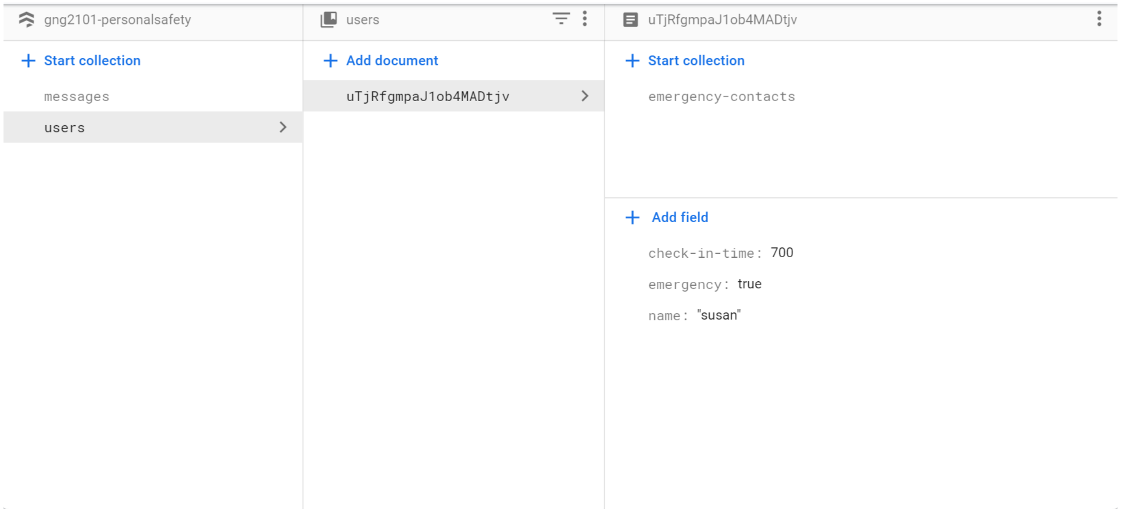
Figure 5. Testing Results for Battery Consumption Plotted



```
29jYr1GhRgiIc9QmLS51
8EuuDSGt6qJ1EwUUg8f5
BTCxun1Pv2XbMCbFVjhE
Qi2Qq14BcfECb6Z1vNAu
by0skizlzyfTAG21J9rP
cHeDt2eVg4WD6V9yt04C
eEjHeHErbPGj9f9rVF21
mWydPjTV67L4i6BLubje
sppQfPvW9d5YQwtNpoyV
sqGr7hIUY0hBycOCimp
v2H20mPtq6IQUdQxI06i
```

```
        dateCreated: March 6, 2022 at 5:08:58 PM UTC-5
        dateSent: null
        dateUpdated: March 6, 2022 at 5:08:58 PM UTC-5
        messageSid: "SMc8aff97655f4423c818d2fc5cf28fdb4"
        messagingServiceSid: null
        numMedia: "0"
        numSegments: "1"
        status: "queued"
    leaseExpireTime: null
    startTime: March 6, 2022 at 5:08:56 PM UTC-5
    state: "SUCCESS"
to: "+16134137365"
```

Figure 6. Messages Firestore Database Collection Organization



messages
users

uTjRfgmpaJ1ob4MADtjv

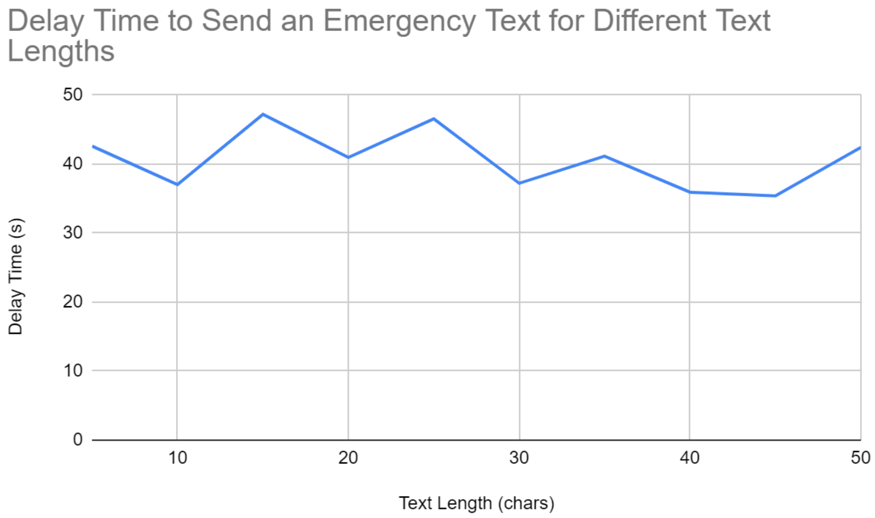
emergency-contacts

check-in-time: 700
emergency: true
name: "susan"

Figure 7. Users Firestore Database Collection Organization

SMS API and Firestore Database Cloud Function Testing

For this test a low fidelity, specific prototype was made to test whether the emergency text length had any impact on the delay time from when the message was sent to when the message was received by the end user. This test was conducted using the Firestore database already created for the app, with the Twilio SMS API extension implemented and a rudimentary cloud function coded on Node.js. The way this prototype works is that in our database there are two collections, messages and users. Under the messages collection, the extension works so that each time a document is created with a “to” (phone number) and “body” field a message will be sent using the contents of the document. The users collection is organized such that each user is assigned a randomly generated ID and has fields for emergency state (true/false), name, emergency contacts and check-in time. The basic cloud function is written so that each time the emergency state of the user is changed, a test text message will be sent to a verified tester’s phone number. Messages of different lengths were used as the test text and the time was measured between when the user’s state was updated to the time the text was received, then plotted below for analysis. As seen in the graph, there is no correlation between text length and delay time. This allows us to create a new target specification for the maximum length of the emergency text. No maximum is imposed by considerations for delay time, but a maximum of 160 characters is imposed by the SMS system. If the message exceeds 160 characters it will be divided into multiple messages which will increase costs related to the Twilio and Firebase services.



c. New target specifications (David)

Table 2 : Updated target specifications

Metrics	Relations (=,<,>)	Values	Units	Achieved (Y/N)/Changed
Notification acknowledged	N/A	Yes	N/A	Changed*
Automatic daily reset	=	1	N/A	N
User Custom check-in time	>	1	N/A	Y
Fail Safe check frequency	=	3	Number	N
The app requires a phrase to be typed when a check is missed	=	1	N/A	N
App communication with preset contacts	N/A	Yes	N/A	N
Emergency contacts Stored	>	3	Number	N
Modes of Communication with emergency contacts	=	2	Number	N
User custom app activation/deactivation (Pause Check in)	N/A	Yes	N/A	N
The app can be set to check multiple times in a day for high-risk activities	N/A	Yes	N/A	N

Number of activities to be selected by user	=	1	Number	N
iOS Functionality	N/A	Yes	N/A	Y
User devices that can sync with app	>	2	Number	N
The app's emergency contacts are prioritized by order and it contacts them in that order, one at a time	N/A	Yes	N/A	N
App functional Languages	>	2	Number	N
User Actions Per Screen	<	3	Number	Y
In-App text Font size	=	20	Points	Y**
The app uses high contrast font/background pairings	N/A	Black/White (and blue)	N/A	Y
User location tracking	N/A	Yes	N/A	N***
Application storage size	<	100	Mb	Y (19.2 Mb for first working prototype)
Battery Consumption	N/A	Minimum	%/min	Y
Emergency Text Length	<	160	Characters	Y

* The initial concept for this functionality was the app being capable of detecting the users actions on their devices, without directly interacting with the app itself. This was changed because it would have been difficult to implement, and the solution devised was more simple, and just as effective.

** The font preset being used is H6. It was selected such that sufficient information can be fit onto any one screen, while still being easily readable by users.

*** It has been decided that this is not a feature that will be implemented. This is due to the amount of work needed to implement it, relative to the unimportance of its function.

4. Conclusion

a. Resume

We received commendable feedback from our client (Susan) with regards to our last meeting with her, and she was visibly pleased with our demonstration, the functionality and user interface design of our prototype. In the document we have covered key takeaways from that meeting as well as our priorities moving on in this project. Our clients feedback serves as confirmation that we are on the right track and this is well received by the team. Various tests were carried out and the results both confirmed certain assumptions as well as provided insights that will guide our next steps as we progress towards the design day.

b. Next steps

Our next step with the back end development will be to improve the cloud functions so that they are higher fidelity. As of right now they will only send test messages that are hard coded, between now and design day the cloud functions must be modified so that they can read data from the Firestore database and use it to send a predetermined emergency message to any phone number listed as an emergency contact by the user. Another functionality that could be added to the back end, depending on time constraints, is multiple user with a registration/log-in page that will validate against credentials stored on the database. This will be required if the app is ever to be deployed to the mass market, but given our time constraints it may be necessary to only have one global user.

The application was successfully uploaded to the App Store of iOS with TestFlight. The local notifications are working and the style and layout of the UI are done.

For next steps we have to implement the screen for the dexterity test and for the sign in. We also need to communicate with the backend server to read and write relevant data, like the emergency contacts list, check-in time, check-in acknowledgement, etc. Finally we will need to make the final touches to the UI/UX to make it as user friendly as possible.

Wrike Snapshot

https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=lTGkJ7eNHzteAR0iCgYXJKhQvnt1l9FB%7CIE2DSNZVHA2DELSTGIYA