GNG2101

Design Project User and Product Manual

SafeTnet

Submitted by:

Personal Safety AND GROUP B14

Sabeeha Ahamed,300167378

Kian Ashrafganjouei, 300173780

Sabina Solomon, 300174303

Darby Martin, 300198642

Iman Jama, 300172218

December 5, 2021

University of Ottawa

Table of Contents

List of Figures

Insert your list of figures here (right-click to update this field).

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
BOM	The Bill of Materials or BOM is a list of materials and their cost.
UI	The UI or User Interface is the place where humans can interact with a program.
UPM	User and Product Manual (UPM) is this report.

Table 2. Glossary

Term	Acronym	Definition
API	Application Programing Interface	Application Programming Interface. APIs allow two applications to communicate with each other.
IDE	Integrated Development Environment	A tool used by programmers to write code efficiently
JS	JavaScript	A programming language often used in mobile and app development

1 Introduction

When developing a project, it is critical to give users the full guide on how to use the product as well as any other important information. The product discussed in this user manual is a mobile application called SafeTnet. This application is on the topic of personal safety. The main objective of this product is to ensure the safety of users by using scheduled check-ins throughout the day. If the user fails to confirm their check-in, their emergency contacts, that they are able to set up, will be notified. This User and Product Manual (UPM) aims to provide the information necessary for clients to effectively use SafeTnet and for prototype documentation.

This guide is divided into sections. The first three parts explain the general information regarding the app such as the overview, the conventions, the cautions, the set-up procedure as well as the system navigation. The fourth part of this UPM goes into detail about the use of the product. It takes the user through every feature of the application and explains how to use them. Also, the technicalities related to the product such as the troubleshooting, the support, the error messages and the maintenance are described in the fifth part of this manual. Furthermore, the sixth section is where the prototype documentation, the bill of materials, the equipment list, the instructions on how to build the product and the testing can be found.

In regards to the assumptions, the first one is that the partnered services that SafeTnet is intertwined with will be operating in the foreseeable future. For instance, Heroku's hosting service is needed to provide the check-in service. Furthermore, it is required that the Apple Store delivers the product to the user. Secondly, it was assumed that the application will attract clients that live alone or like to do activities alone.

As for security and privacy considerations associated with the use of the User and Product Manual, it has been established that there are no concerns.

Finally, the intended audience for this document are the users of the SafeTnet application, the future students who will take on this project and/or for anyone who is interested in learning more about this mobile application.

2 Overview

SafeTnet is an app for anyone who lives alone, enjoys individual activities, has a dangerous job or anyone who isn’t in contact with your loved ones as often as they would like. It allows them to keep their loved ones or emergency contacts updated on their safety. Since everyone lives such busy lives that it's hard to continuously text or call. Receiving constant text while at work, isn’t always an option. SafeTnet ensures that pre-set contacts will be notified automatically only when you could possibly be in danger. The app will keep track of your status through the check-ins you create. Only if you miss a check-in, we will alert your contacts.

The app allows users to create several check-ins to confirm their safety and set up emergency contacts. If the user hasn’t checked in an hour after the check-in time, the app will send a SMS message to their emergency contacts.

Unlike competitors, our app works in Canada, is both iOS and Android compatible, allows the user to create multiple check-ins, and is cheaper than the competition (priced at $12/year).

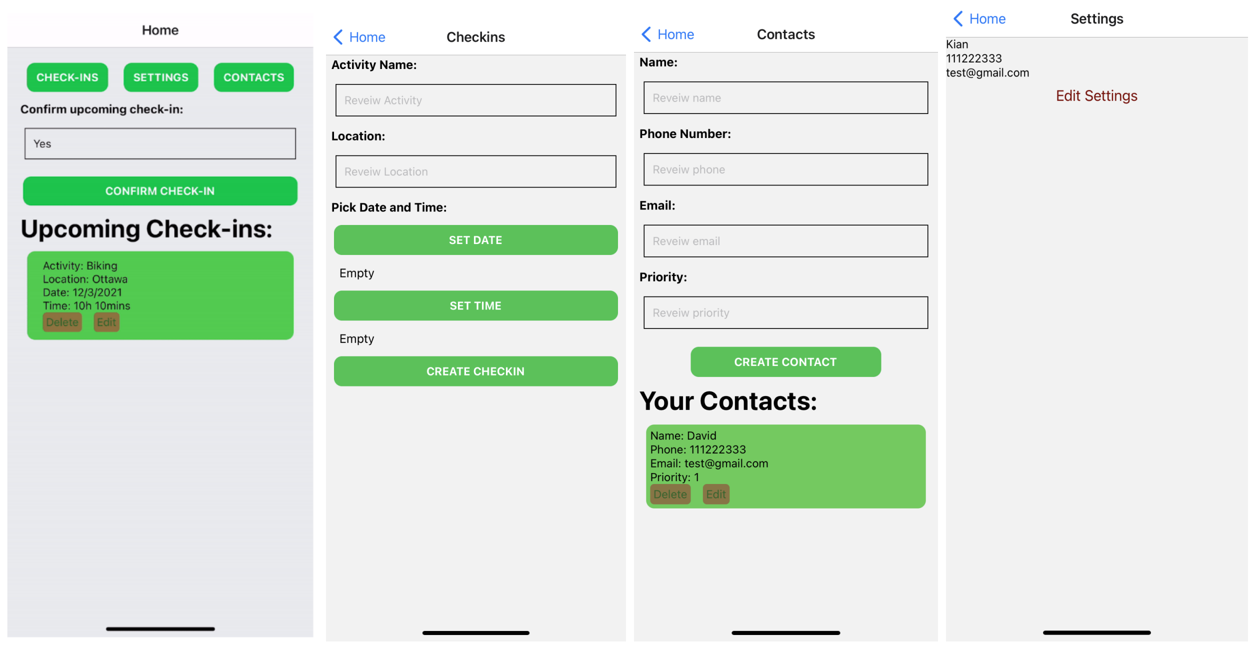


Figure 1 : Screenshots of final prototype.

The app allows users to view, create check-ins, and complete check-ins. Check-ins are times at which the user commits to going back to the app to confirm their safety. If the user fails to confirm their safety within one hour after their check-in time, their emergency contacts will be notified. The confirmation of check-ins is done on the home screen, by typing “yes” and tapping “CONFIRM CHECK-IN”. For more information about how to create check-ins and complete check-ins within the app please view Section 4.1 and 4.4.

The user can also view and create emergency contact’s information. These are the people who will be contacted once a check-in is missed. For more information please see Section 4.2.

Finally, the user can access their Settings and update their personal information. For more information please see Section 4.3.

The application is a mobile application. It is JavaScript based. The frontend is built using React Native framework and the backend is built with Node.js. The backend is hosted on Heroku. Firebase Realtime Database is used as the app’s database. SafeTnet also uses Twilio API to send SMS messages to the emergency contact.

The user interacts with the application through buttons and text fields designed on the front end. The text fields have hints to make the expected input from the user clear and improve usability.

2.1 Conventions

The app’s theme is green. Buttons, Checkin, and Contact components all use #52cc50 color hex code.

All button labels are in all capital letters to make them easier to identify.

Hints are used in text fields to make the expected input from the user clear.

2.2 Cautions & Warnings

At the time that this manual was completed, the app was not fully functional. Please do not use the current version of the app for personal safety.

The user manual will be updated as we further as the development process continues.

3 Getting started

The app is not yet fully developed. Once it is, it will be published in the app stores found in all mobile phones. Right now, you can run the app only by running the programming files themselves. You can access those files through the GitHub repository for the app. The link can be found in the Appendix of this user manual. To set up your devices to run the app, follow the steps mentioned in Set-up Considerations.

Once you have the app running, you will first see the home page where you can view all the check-ins. There is also a navigation bar where you can switch between different pages of the app. Each page has its own functionality. The functionalities of the app will be explained in more details later in the manual.

3.1 Set-up Considerations

You would need a desktop computer or a laptop to run the programming files. At the moment, there are two ways of running the app using a simulator - running it on a IOS device or running it on an Android device. The procedure is mostly the same but depending on the device you are using, you may need to install different things.

You need to download a few things first before you can get started:

1. Download a text editor to run the programming files. A text editor is a software you can use to edit plain text. A common text editor is Notepad which can be found in any Windows computer. You can use any text editor, but Visual Studio Code is highly recommended.
2. You need to go to your Command line (called Terminal in macOS) to install software packages needed to run the application. Make sure you have Node.js installed. If not, then you can search online to download the latest version. Then use the command “npm install --global expo-cli” . This downloads expo packages which will be used to run the app on a simulator.
3. If you are using a Mac then make sure you have Xcode installed properly. If you have a Windows computer, you will need to download the Expo mobile application on a separate mobile device.

3.2 User Access Considerations

The app is available to both IOS and Android users. However, all the users must have access to the internet in order to use the application.

3.3 Accessing the System

Once you download the files from the repository and all the others mentioned above, you are ready to start running the applications. Open the files you downloaded in the text editor. The steps outlined here are from using Visual Studio Code.

From the navigation bar, select Terminal -> New Terminal. This will open a Terminal window at the bottom of your text editor. In the terminal enter the command “npm start”. It should open a window on your web browser. If you are using a Mac, then you can go back to the terminal in the text editor and type “i”. This will open a simulator using Xcode. If you are not using a Mac, scan the QR code in the web browser using your phone. It will simulate the app in the expo app you downloaded earlier. It may take a few seconds to load. Now you can use the SafeTnet app.

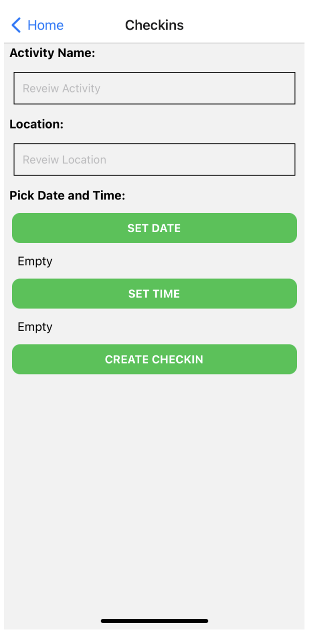
3.4 System Organization & Navigation

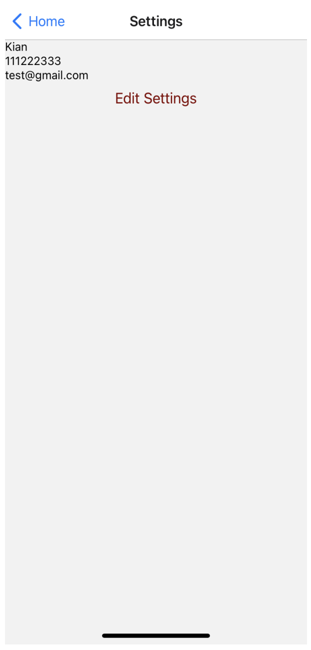
The home page is where you can view all the check-ins that have been created. Each check-in shows the activity, location, date and time. There is also a navigation bar where you can go to different pages: Check-ins, Settings and Contacts.

The “Check-ins” page allows you to create a new check-in. You can add the necessary information needed to create one. Once you click on the “Create Check-in” button, you click “Home” to go back to the home page where you can view the newly created check-in.

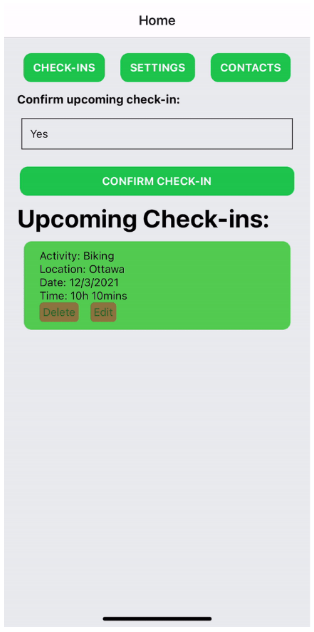
The “Settings” page is where you view your own information. This information can be updated if you click on the “Edit Settings” button.

The “Contacts" page is where you can view your emergency contacts. You can also create new contacts by inputting all the info and clicking on the “Create Contact” button.

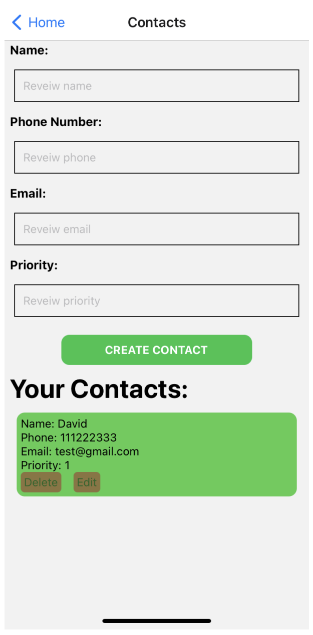




Kian
111222333
test@gmail.com



Activity: Biking
Location: Ottawa
Date: 12/3/2021
Time: 10h 10mins



Name: David
Phone: 111222333
Email: test@gmail.com
Priority: 1

Figure 2: The four pages of the application.

3.5 Exiting the System

You must first close the simulator you are running the app on. If you are using the Expo app on your mobile application, then you can just close the app. If you are using the Xcode simulator on a Mac, then close the simulator. Then you must kill the terminal of the text editor. This will disconnect you from Expo CLI. You can now close the web browser and text editor.

4 Using the System

In this section, the specific functionalities of the app are described in depth.

4.1 Schedule Check-ins

To schedule a new check-in, you need to first enter the “check-ins” screen. Then you must enter the information for the check-in. Firstly, there is the “Activity” field. This is where you enter the name of the activity that you are doing when the check-in comes. Then there is the “Location” field, this is where you enter the location you will be in when the activity comes. Next you must enter the date and time of the activity -- see below for details. Once the check-in is complete, you must click on “create check-in” to complete the process. You can now go back to the home screen.

4.1.1 Set the date and time of check-in

To enter the date of the check-in, you must first click on the “add date” button, a pop up date field will appear on the screen, click on the popup to open the calendar. Once the calendar is open you can choose the desired date and exit the calendar by clicking anywhere on the screen other than the calendar. Next, you can click on the “add time” button, a pop up time field will appear on the screen, click on the pop up to open the time picker widget. Once the time picker is open you click anywhere on the screen other than the time picker to close the widget. You have now completed the process of creating a new time and date for the check-in.

4.2 Create Contacts

To create contacts, you must first enter the “contacts” screen. Here, there are several fields to complete for the information of the contact. First, there is the “Name” field. Next there is the “Phone Number” field. Next, you must enter the email address of the contact in the “Email” field. Lastly, there is the “Priority” field which is used to indicate which contact would be contacted first in the case of a failed check-in. Lower numbers indicate a higher priority of contact (e.x. a contact with priority of“1” would be the first person to be contacted).

4.3 Update Settings

To update the settings, you can enter the “settings” page. update your name, email, and phone number by entering them in the “Name”, “Email”, and “Phone number” fields respectively. Once the information is entered, you can click on update settings.

4.4 Confirm Check-in

To confirm a check-in, you must first wait for the check-in time and date to be reached. Then you can go on the "home" screen to confirm the check-in. The first step is to enter "yes" in the "confirm upcoming check-in" field. Once this is done, you can click on "confirm check-in" to complete the upcoming check-in. The upcoming check-in will be removed from the list of upcoming check-ins and the check-in confirmation will be completed.

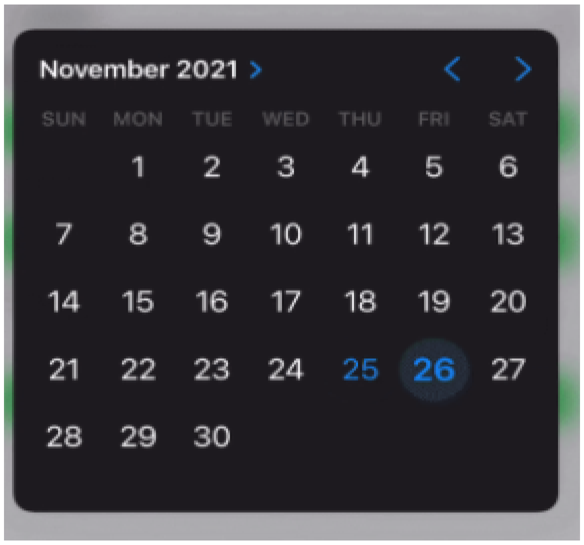


Figure 3: Picking a date

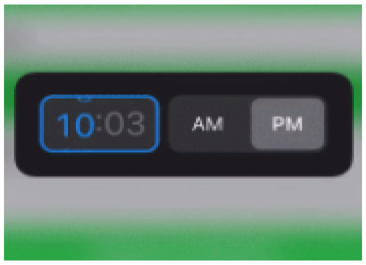
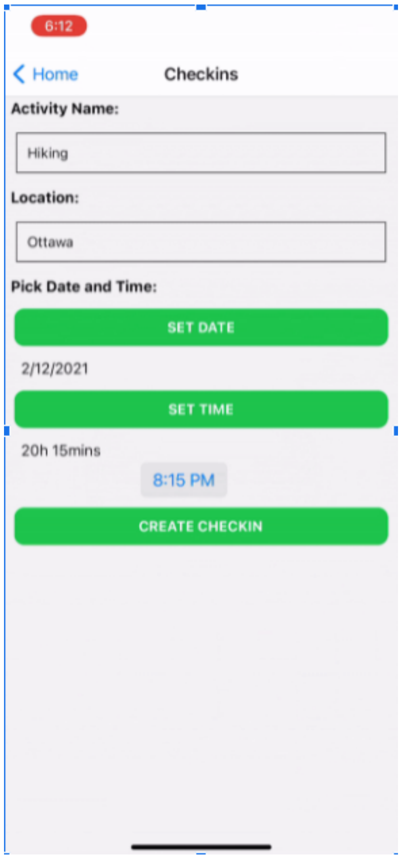


Figure 4: Picking a time



Activity Name: Hiking

Location: Ottawa

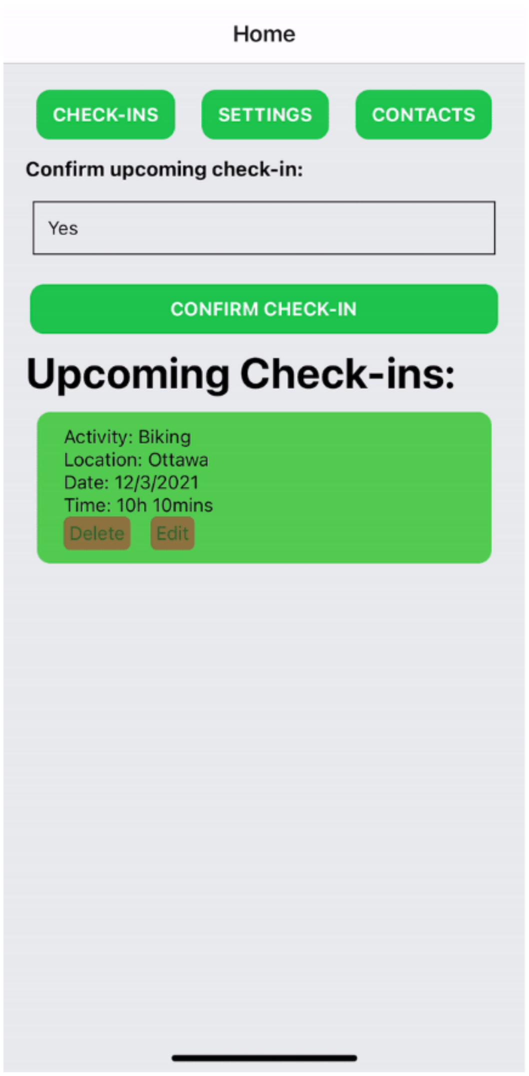
Pick Date and Time:

2/12/2021

20h 15mins

8:15 PM

Figure 5: Creating check-ins



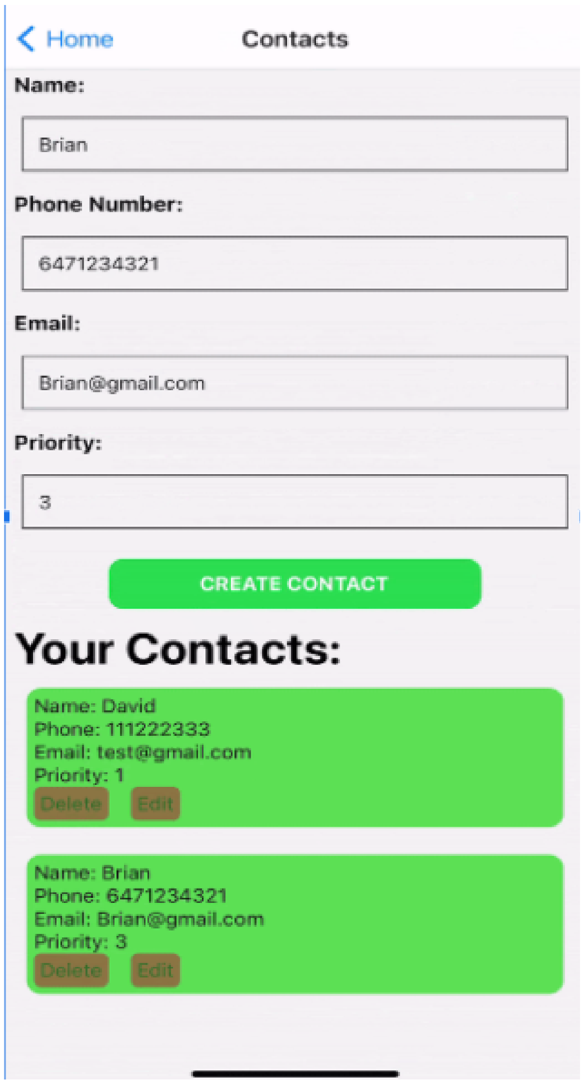
Confirm upcoming check-in:

Yes

Upcoming Check-ins:

Activity: Biking
Location: Ottawa
Date: 12/3/2021
Time: 10h 10mins

Figure 6: Confirming check-ins



Name:

Brian

Phone Number:

6471234321

Email:

Brian@gmail.com

Priority:

3

Your Contacts:

Name: David
Phone: 111222333
Email: test@gmail.com
Priority: 1

Name: Brian
Phone: 6471234321
Email: Brian@gmail.com
Priority: 3

Figure 7: Creating contacts

5 Troubleshooting & Support

In this section, a list of known errors are provided. Furthermore, the contact information for the developers of the project in case of assistance are provided.

5.1 Error Messages or Behaviors

List of errors known:

- Sometimes, after creating a new check-ins, a warning appears on the bottom of the screen saying that “Warning: Encountered two children with the same key”.
- After creating a new check-in, the order of the displayed upcoming check-ins in the home page may change.
- It is possible to create check-ins without implementing all of the fields or implementing the fields correctly, for example, if you attempt to create a check-in without setting the date and time, those fields will appear as empty.
- The delete and edit buttons have not been implemented yet for the contacts and check-ins.
- Check-ins and contacts created within the app will not appear if you close the app and open it again. They are currently being saved in the database. However, they are never being fetched.

5.2 Support

Please contact the developers of the project through email.

- Kian Ashrafganjouei: kashr085@uottawa.ca
- Darby Martin: dmart141@uottawa.ca
- Sabina Solomon: ssolo008@uottawa.ca
- Sabeeha Ahamed: saham083@uottawa.ca
- Iman Jama: ijama023@uottawa.ca

6 Product Documentation

6.1 Software

6.1.1 BOM (Bill of Materials)

Twilio - https://www.twilio.com/sms/pricing/ca

Firebase Firestore database - https://firebase.google.com/pricing

Heroku - https://www.heroku.com/pricing

6.1.2 Equipment list

- A computer with Node.js and ReactNative
- A code editor or IDE such as VScode is highly recommended
- A simulator such as Xcode on Mac or Expo on your ios device is highly recommended
- Firebase Database is recommended however, any database compatible with Node.js will work
- Heroku is recommended however, any hosting platform compatible with Node.js will work
- Twilio is recommended however, any SMS service compatible with Node.js will work

6.1.3 Instructions

Creating the server side of the application can be done using the Node.js Express library. This can be installed using the command “npm install express” on the command line when in the project directory. You can use Express to listen for incoming HTTP POST requests. You must handle each possible request that can be sent from the application. Setting up a database to store the data being received can be done using Firebase. Because this is a web server, the Firebase

Admin module must be used. This can be installed using “npm install firebase-admin”. Once the Firebase Admin is set up you must create an app on the firebase console and link it to your existing Node.JS application. This can be done by following the Firebase setup guide on https://firebase.google.com/docs/admin/setup. Once the HTTP and database functionalities are set up you can use the Node-schedule library to keep track of the check-ins. Node-schedule can be installed using “npm install node-schedule”. Node-schedule will perform a predetermined task at a specific time. This can be used to schedule check-ins and to initiate sending out a text to emergency contacts. Once the scheduler determines that it is time to send out a text, Twilio is used to actually send out the SMS messages. Twilio will need to be installed with “npm install twilio”. You can create an account with twilio and purchase a virtual phone number. you can link it to your code using the Twilio documentation found at https://www.twilio.com/docs/sms. Once Twilio is set up, you will be able to send out text messages from your code. The last step in creating the backend of the application is to host your code. By setting up an app with heroku, you will be able to remotely host your code. Heroku can be set up from the command line or through your github repository. A thorough guide on deploying your code on Heroku can be found at https://devcenter.heroku.com/articles/deploying-nodejs. You now have the server up and running.

Creating the client side of the application can be done using the React Native library. This can be installed using the command “npm install react-native-app” on the command line when in the project directory. You would also need to install the expo through “npm install --global expo-cli” on the command line. Next you open up the directory where the react native application was installed and implement the application code. Once the coding process is complete, you need to run the command “expo start” on the command line. Next, a window on your default browser will open providing you with the option to simulate the application either on a virtual mobile or through the exop app on a mobile device.

6.2 Testing & Validation

Testing was done by running the application on a simulator and monitoring how it reacted to user input. Alternate testing on the code was done using debuggers and code output statements. Pictures of the running application can be seen above in section 4.

7 Conclusions and Recommendations for Future Work

Building a functioning application is quite challenging. It requires a lot of research especially if you as the development team have little to no experience working with the language or tools that you will be using. The research required to figure out how to complete a project such as this one takes up the majority of time. Unfortunately, there is little time left to get used to the syntax of the new language or the libraries and other tools being used. It would be beneficial to use as many tools that you are already familiar with to decrease the amount of time it takes to research.

Another critical lesson that we've learned is that we need to leverage modern development tools that simplify the programming process. These tools can come in a variety of forms, such as Git which helps with version control for our application and maintain a remote repository. There is also Figma, which allows us to design the user interface before the coding process. We've also used debuggers and testers to solve errors and find potential bugs ahead of time.

If we had more time to work on this project, there are a couple functionalities that we would like to add. Allowing the server to send notifications to the user when it is time to check in would be a high priority. Other desired functionalities would include: updating all elements of the user interface with data from the database, allowing the server to retrieve the user's location and user authentication. Also, if we had more time, we would continue to improve the user interface.

8 Bibliography

- Heroku: https: www.heroku.com
- React native: www.reactnative.dev
- Node.js: https://nodejs.org/en/
- Figma: https://www.figma.com/
- Expo: https://expo.dev/

APPENDICES

9 APPENDIX I: Design Files

These other files are an archive of our progress and documentation throughout the development process of this project. PDB contains the problem statement, metrics. benchmarking and target specifications. PDC contains the project plan, the initial designs and the feasibility study. PDD contains our detailed designs, prototype 1 and the BOM. PDF contains our prototype 2. PDG contains the business model and economics report.

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
PDB	https://makerepo.com/KianAshrafganjouei/1077.safetnet-b14	Sep 23, 2021
PDC	https://makerepo.com/KianAshrafganjouei/1077.safetnet-b14	Sep 30, 2021
PDD	https://makerepo.com/KianAshrafganjouei/1077.safetnet-b14	Oct 7, 2021
PDF	https://makerepo.com/KianAshrafganjouei/1077.safetnet-b14	Nov 11, 2021
PDG	https://makerepo.com/KianAshrafganjouei/1077.safetnet-b14	Nov 18, 2021

10 APPENDIX II: Other Appendices

Link to github (front end): https://github.com/KianAshrafganjouei/gng2101-protoype2

Link to github (back end): https://github.com/dbjmt107/gng2101-Server-Side

Link to Figma (used to design UI):
https://www.figma.com/file/eZRswKX4QioMcp3xpHsd4L/Prototype