**Final Test Report**

**Running Safety - Team 22:**

Thomas Cimino

Zexing Gao

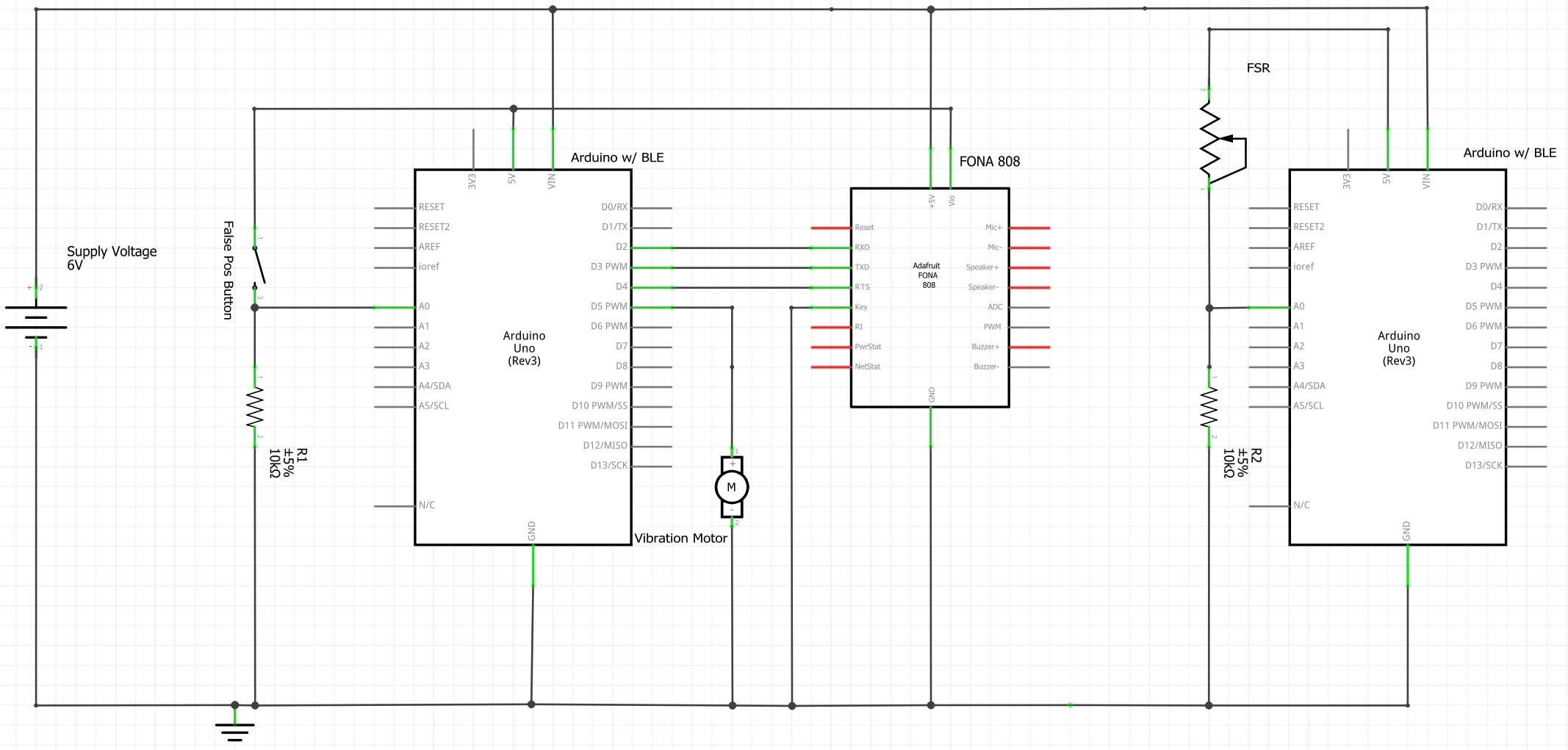
Cong Han

Payton Hauck

Yajing Lai

Hardware Setup:

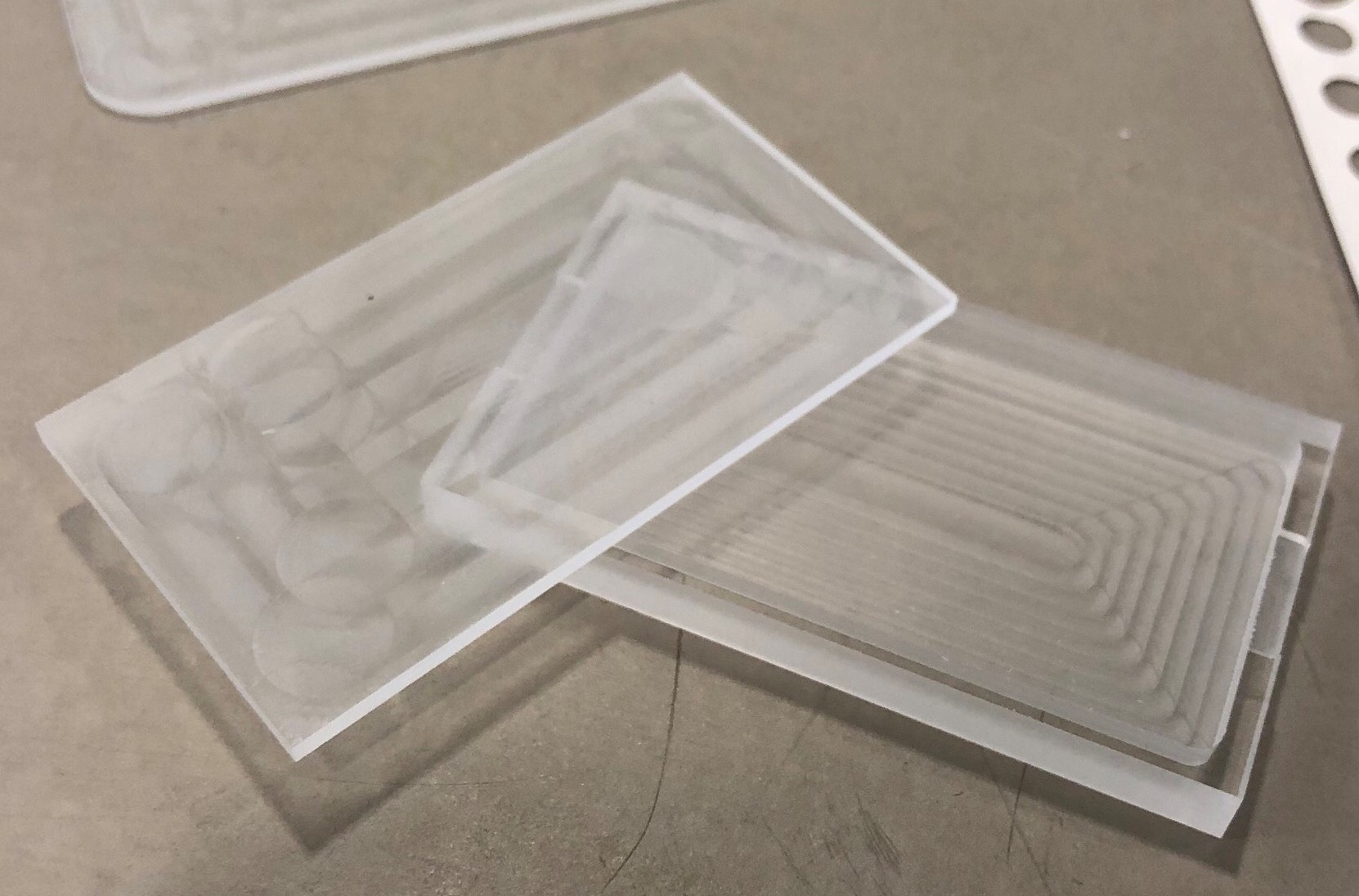
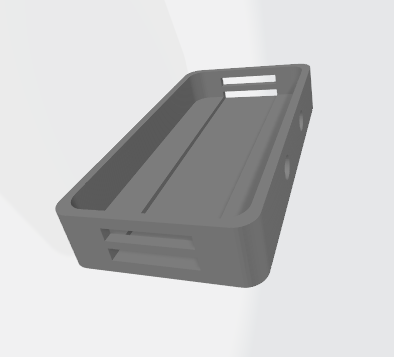
Our hardware setup will consist of the two main device components, each showing distinct functionalities of our product. There are two isolated Beetle BLEs connected to each other via bluetooth. The shoe component consists of the slave Beetle, which will be connected to a force-sensitive resistor. The chest strap component consists of the master Beetle which is connected to our GPS/GSM module, pulse monitor, vibration motor, and a push-button.



\*Pulse monitor not shown (No Fritzing model)

What to perform for hardware tests:

The test will show the autonomy of our design. We will turn on the two sub-devices and they will connect automatically. From there we will run several trials. The first will be repeated, frequent pressure events on the force-sensitive resistor to show that the device remains in standby mode, simulating a jogger’s repeated pressure on his or her feet. The second trial will show the system automatically responding when there is no pressure on the force sensor. After ten seconds of no pressure, the shoe insert will automatically send a signal to the main chest strap module with the vibration motor. The vibration motor will turn on for another ten seconds. After that, the chest strap module will send a signal back to the shoe insert module, which will put that shoe insert the module into permanent standby. The chest strap module (connected to the GPS/GSM board) will enter distress mode, sending the user’s Google Maps location via Twilio to the user’s emergency contact that is in the device database. The third trial will show the functionality of the false-positive cancel button. Again, we will let the shoe insert timeout, sending the alert signal to the chest strap which turns on the vibration motor. This time, however, we will press the cancel button on the chest strap, which will turn the vibration motor off, and reset both devices to normal. We will go on in this trial to let the device automatically respond, to show that they are constantly monitoring their environment even after a false-positive cancellation.





Hardware Measurable Criteria:

1. No false detections under normal operation on pressure sensor (no warning signs)
2. Total system timeout and emergency text sent after 20 consecutive inactive seconds (vibration motor on)
3. Warning of system timeout after 10 consecutive inactive seconds (vibration motor on)
4. Return to normal operation after warning cancellation (no vibration warning)
5. Total system timeout and emergency text sent after 20 consecutive inactive seconds **after** warning cancellation

Hardware Results:

1. No false detections occurred.
2. System timed out as it’s supposed to.
3. Warning occurred as it’s supposed to.
4. The system did return to normal operation after warning cancellation.
5. The system correctly timed out when it was supposed to after warning cancellation.

Hardware Conclusion:

Our hardware systems are working exactly as they should be. Professor Pisano specifically liked how well the vibration motor worked and the option of the cancel button. A few more run through tests will be ran in order to make sure the device is working fully before we give it to our customer. We have all pieces in the enclosings, but now need to seal the device and cover the shoe insert. Our final test will using the device ourselves during a test run next week to ensure all hardware survives normal exercise. Then our customer will be given our device.

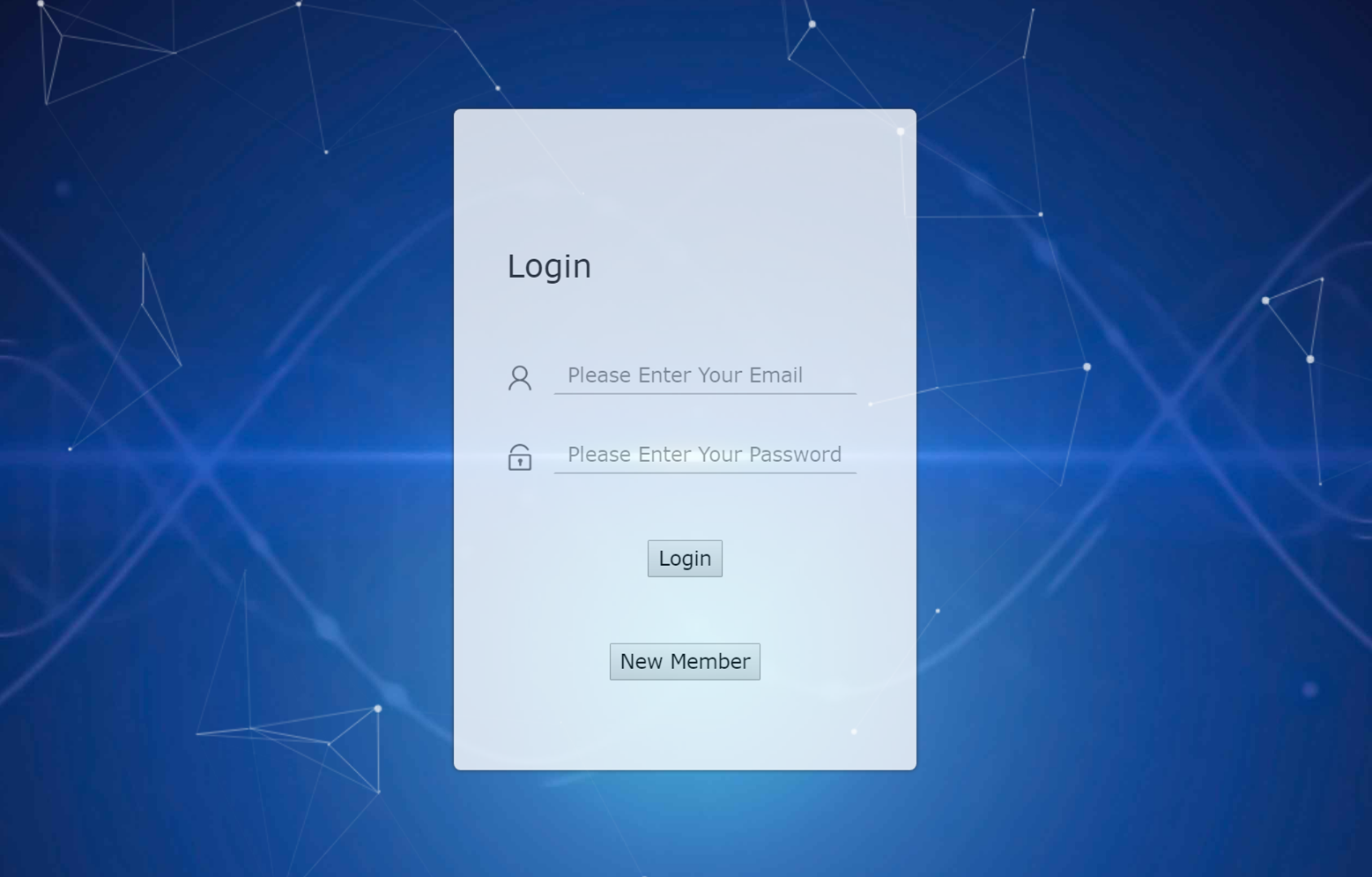
Software Summary:

We used an internet connected laptop with Windows OS to demonstrate our software project. The user interface, using JavaScript and node.js, is tested on Chrome via Webstorm IDE. We have successfully implemented all major requirements for the software including login and sign-up page, along with the home page that shows location and requirements data entering. All functions described in our test plan have been successfully tested and demonstrated.

Software Test Process:

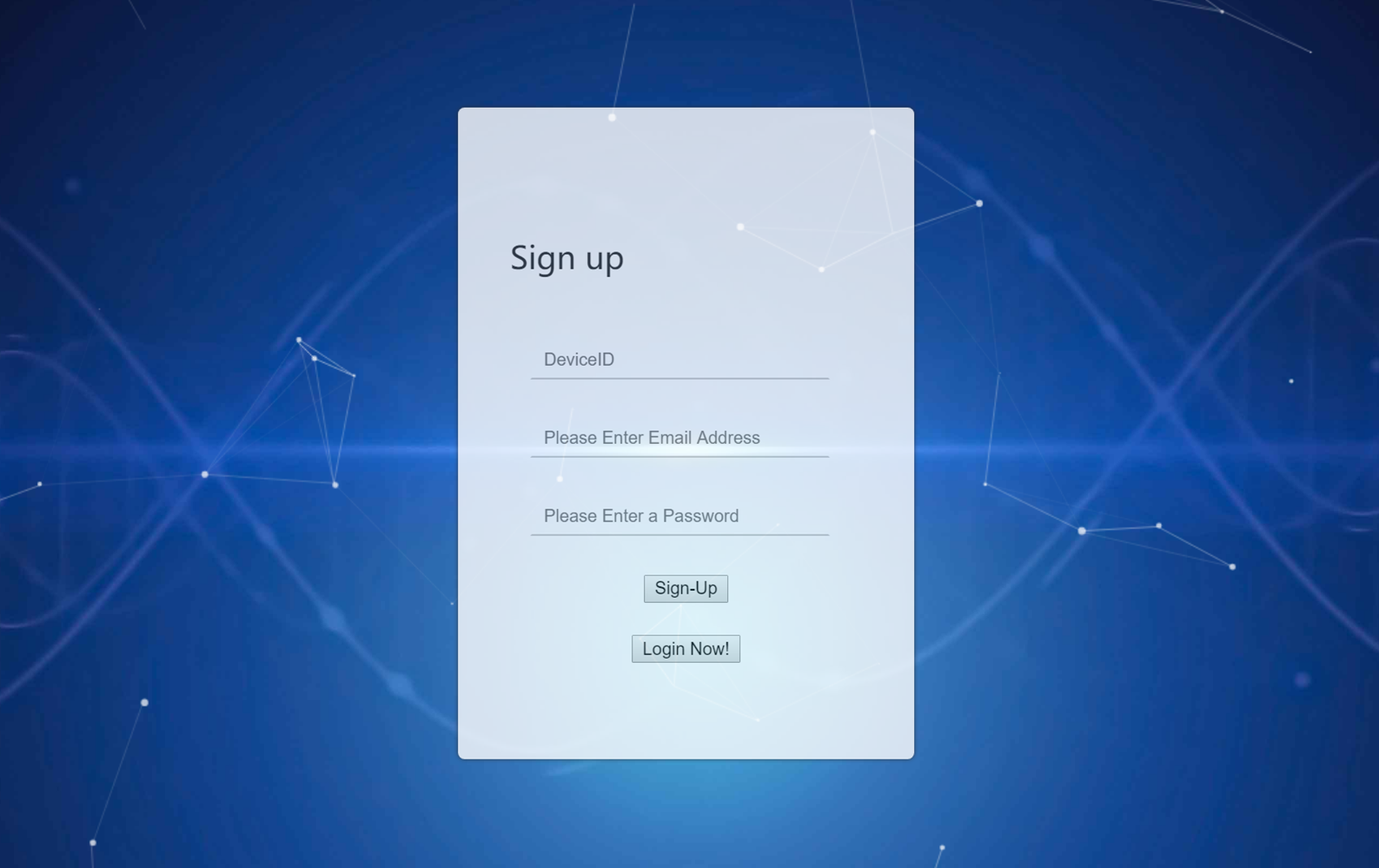
1. *Showed login & registration interface.*

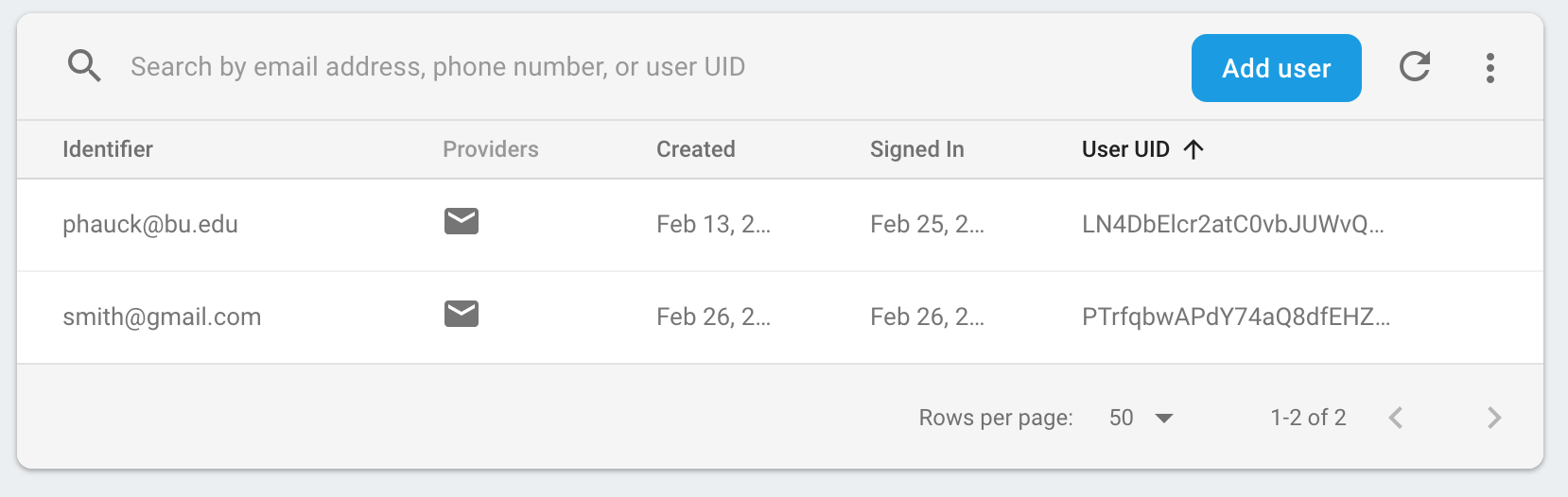
We used HTML, CSS, and JavaScript to design the user interface. Our CSS styling still could use a few improvements but we would save those small details for version 2.0.



1. *Showed login & registration backend method.*

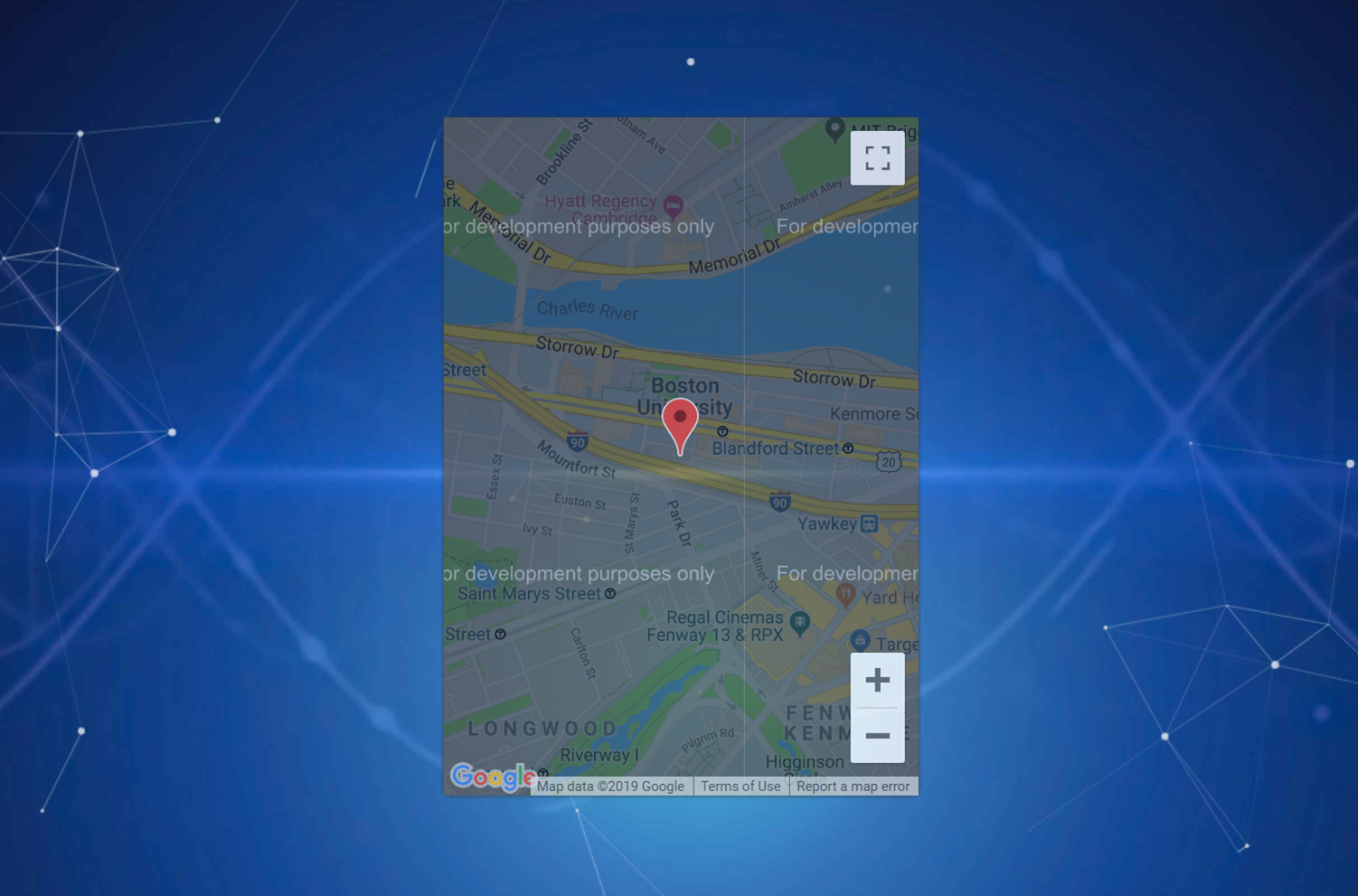
Firebase was used entirely for the authentication process with our web app. We also will be using firebase for database storage to keep it organized and simple. Users can log in easily or choose to signup with they use our web app. All authentication information is stored in the Firebase system that administers can make modifications on those accounts if needed.





1. *Showed map function with current location*

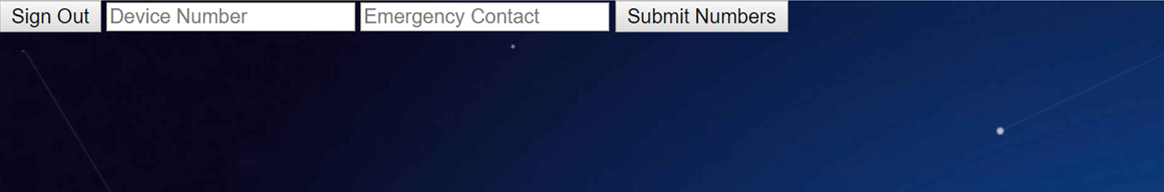
Here we used Google Map API. It is the most common reference as a map API, and it is also the most powerful API. It can help to solve all of our requirements. In the text message, it will include the geographic coordinates, and we will put these coordinates into our web app to plot the graph.



1. *Showed Registration of Device and Contact input*

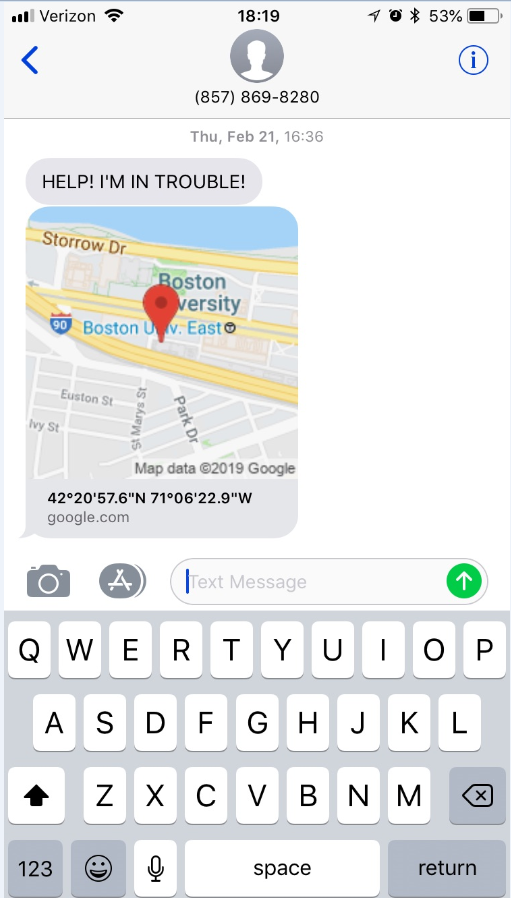
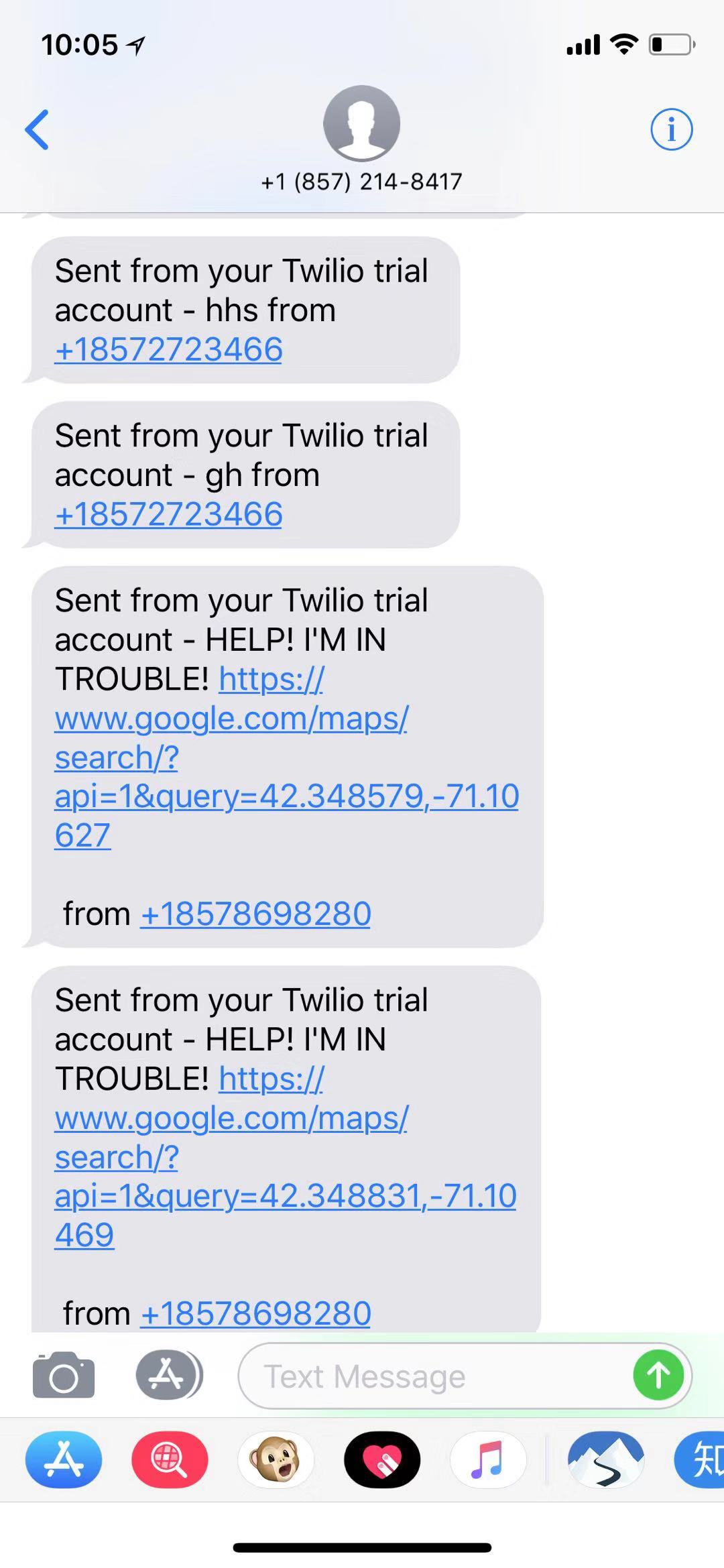
On the home page, the user is also able to enter the device number and emergency contact that they would like to use with their device. The user can do that by simply click the register button and input the device id and emergency contact into the textbox. The device number and contact number are linked to each other so the hardware knows what devices are connected to which contacts. All are stored in the cloud database. When the device sends out the text message, it will include the device ID. Therefore we can track the ID to determine which contact number belong to this device.





*5. Location message via Twilio*

The user can receive a text message with a link that we can open on google map that will show the exact location of the runner (instead of Lat/Long coordinates). In short, we have three numbers in total: the hardware device number, Twilio cloud virtual number, and the emergency contact number(s). Twilio is the bridge to communicate between our device and emergency contacts.

*6. Deploy web app on AWS*

We use AWS to create a server which we run our web app on this server. Since the IP address will change every time, so we use the elastic IP address which will not change. Then we associate the elastic IP address with the virtual machine address then link our domain name to this IP address. Our domain name is [www.seniordesignrunningsafety.com:8080/login.html](http://www.seniordesignrunningsafety.com:8080/login.html)

Measurements of success

1. We successfully opened the webpage and finished the sign-up process by using the information which the professor randomly gave.
2. The website is successfully opened via the access of the public domain [www.seniordesignrunningsafety.com:8080/login.html](http://www.seniordesignrunningsafety.com:8080/login.html). And the same link can be open on any users’ phone.
3. The user’s location (last updated information) can be found on the map, and shown properly to the user.
4. The given authentication information was updated into our firebase system, which shows that the authentication information we just signed up has been updated into the cloud.
5. After the SMS message sent from our hardware device, our Twilio virtual phone number received that information and forward it to Professor Pisano’s phone number with “need help” information and correct GPS location information. The GPS location is successfully opened with Google map on the user’s phone. Meanwhile, the geographic location has been updated into the webpage, so that we can see their current location.

Software Conclusion:

In short, we achieved all of our goals that were set at the beginning of the semester and the measurements set in the final test plan. The software test was completed successfully but there was a delay in receiving the message. Professor Pisano registered with his phone number and he received the emergency text as expected but there were a few glitches along the way, resulting in a delay. We will fix this delay so that it does not occur for our customer during the installation period. Other than the delay all software tests succeeded and the AWS domain is also up and running. We can now move into the next stage: customer installation.