**2nd Prototype Test Report**

**Running Safety - Team 22:**

Thomas Cimino

Zexing Gao

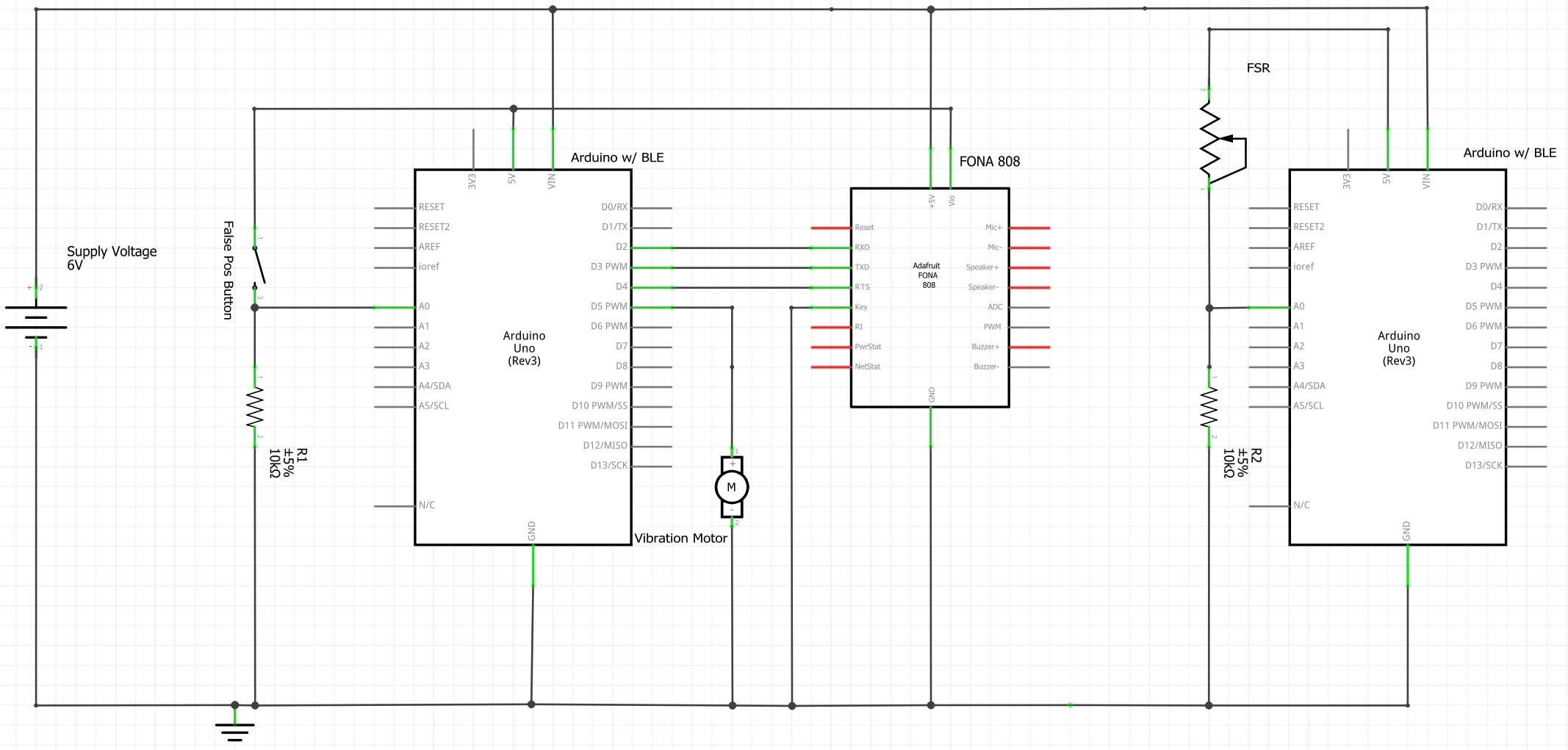
Cong Han

Payton Hauck

Yajing Lai

Hardware Setup:

Our hardware setup will consist of two main breadboards, each showing distinct functionalities of our product. On one breadboard, there will be two isolated Beetle BLEs connected to each other. A force-sensitive resistor will be connected to the slave Beetle and a vibration motor and push-button will be connected to the master Beetle. The master Beetle will also be connected to our GPS/GSM module, pulse monitor, and a push-button.



What to perform for hardware tests:

1. The first test will show the autonomy of our design. We will turn on the Beetles, let them connect automatically, and 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 Beetle connected to the force sensor will automatically send a signal to the Beetle with the vibration motor. The vibration motor and the Beetle’s on-board LED will turn on for another ten seconds. After that, the Beetle controlling the vibration motor will send a signal back to the Beetle on the force sensor, which will turn on that Beetle’s on-board LED, signifying that Beetle has entered permanent standby. The master Beetle (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. The third trial will show the functionality of the false-positive cancel button. Again, we will let the first Beetle run to timeout, sending the alert signal to the second Beetle to turn on the vibration motor. This time, however, we will press the button connected to that Beetle, which will turn the on-board LED and vibration motor off, and reset the first Beetle to normal. We will go on in this trial to let the Beetles run until automatic distress response, 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 and both LEDs on)
3. Warning of system timeout after 10 consecutive inactive seconds (vibration motor and one LED on)
4. Return to normal operation after warning cancellation (all warning signs turned off)
5. Total system timeout and emergency text sent after 20 consecutive inactive seconds **after** warning cancellation (see Item 2)

Hardware Results:

1. No false detections occurred.
2. System timed out as it is supposed to.
3. Warning occurred as it is 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 as they should so far. Each measurement criteria was addressed and met. The next steps will be mounting the components into their respective housings: the chest/back housing and Dr. Scholl’s insert. After that, we’ll do Q.A. tests, write user instruction manuals, and then be done with the hardware.

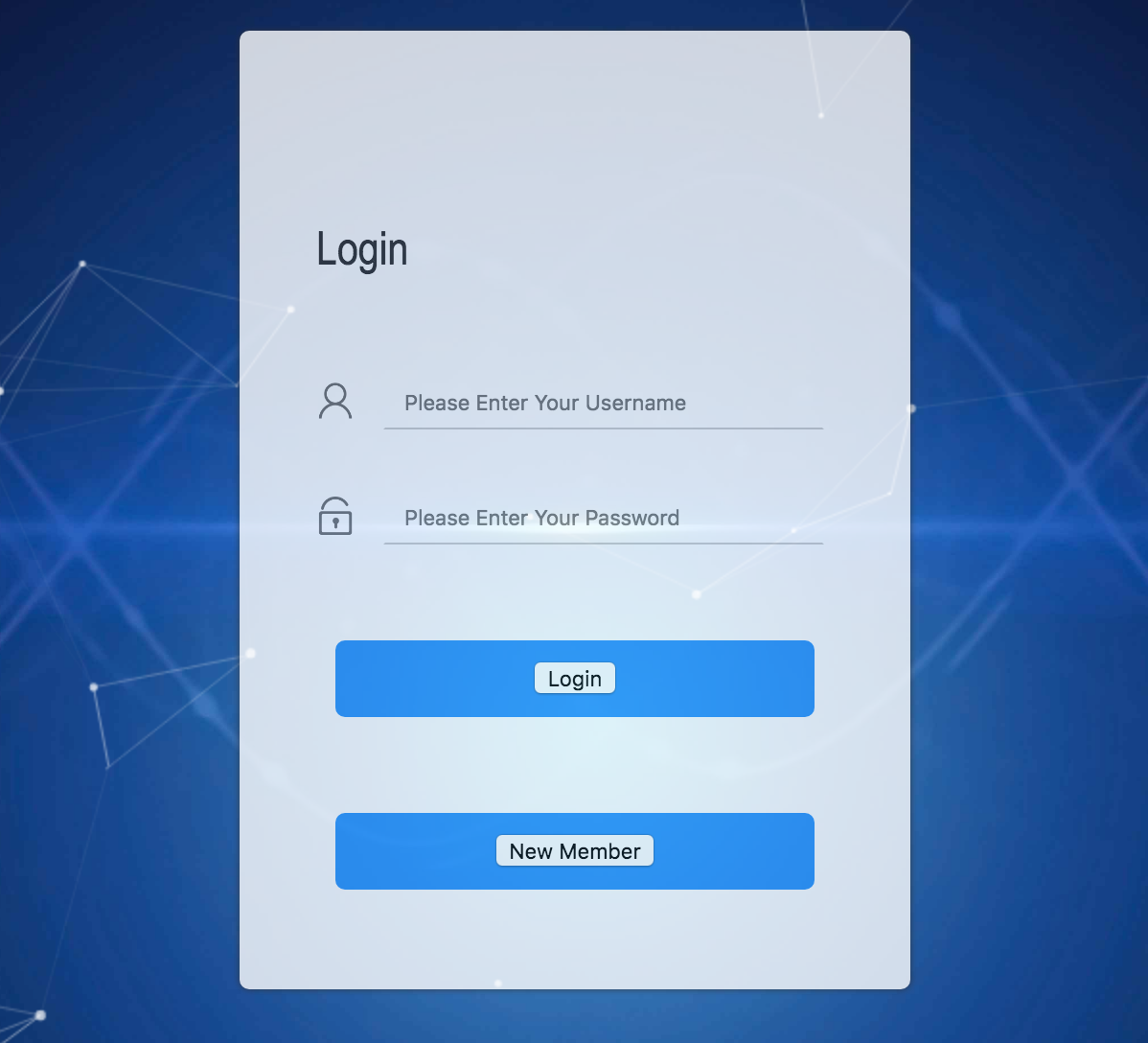
Software Summary:

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 the majority of the user interface qualities, including login and sign-up page along with a home page displaying the user’s location. All functions described in our test plan have been successfully tested and demonstrated. There are still a few improvements that will need to be made before our final test.

Software Test Process:

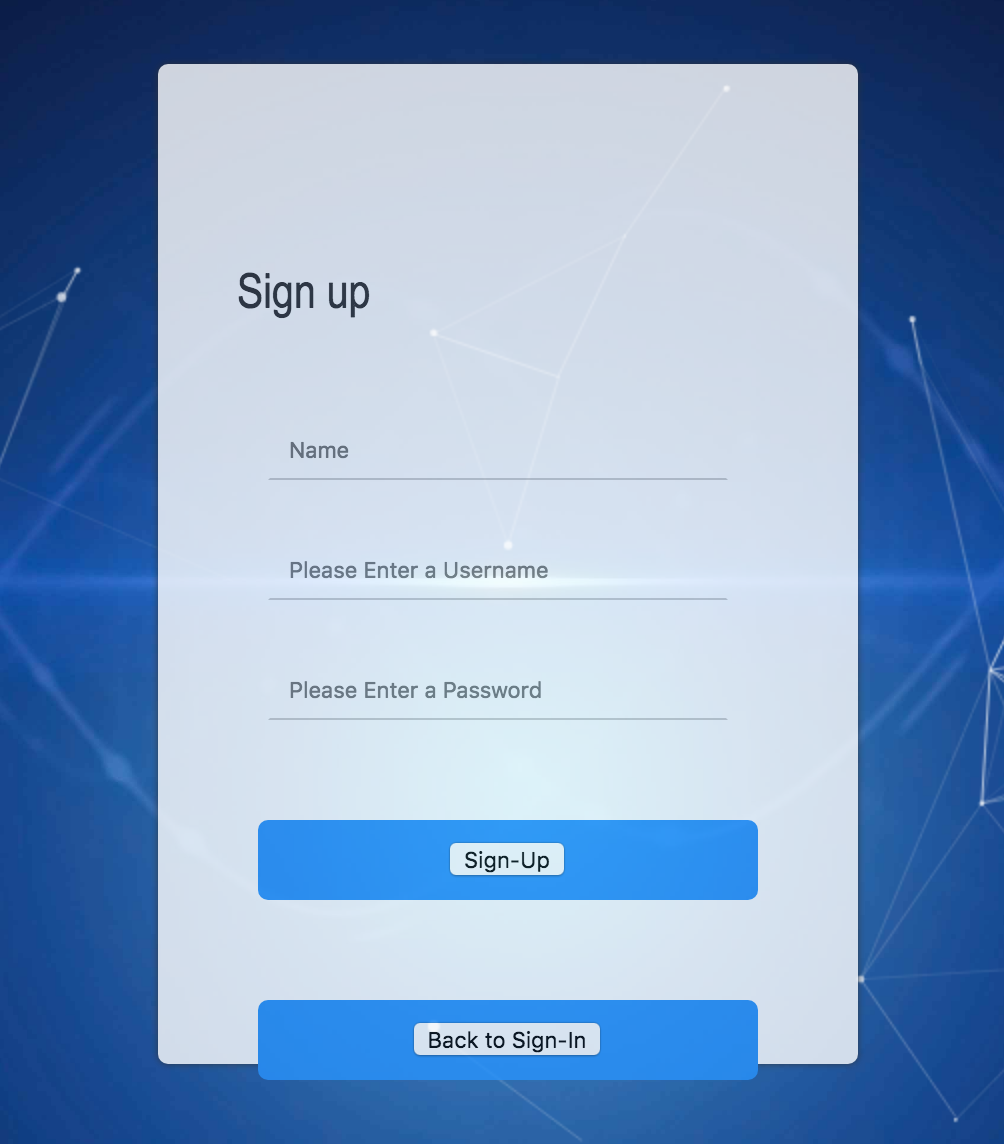
1. *Showed login & registration interface.*

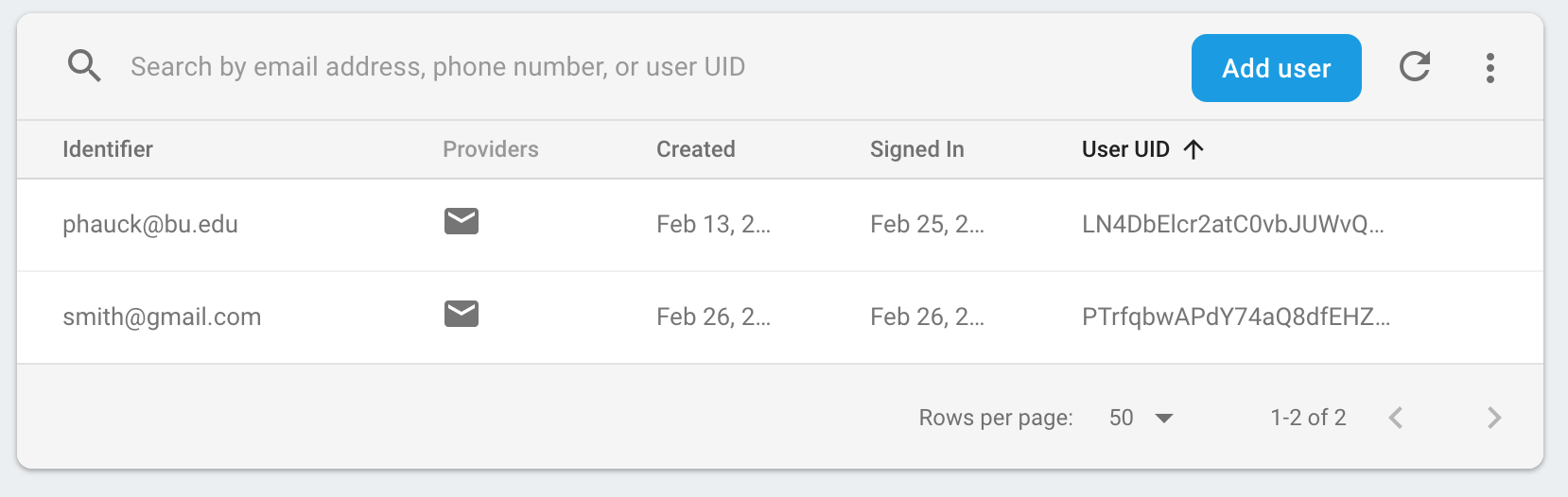
We used HTML, CSS, and JavaScript to design the user interface. Some CSS styling will need to be handled after we have a fully functional app, but this will be our last step for software development.



1. *Showed login & registration backend method.*

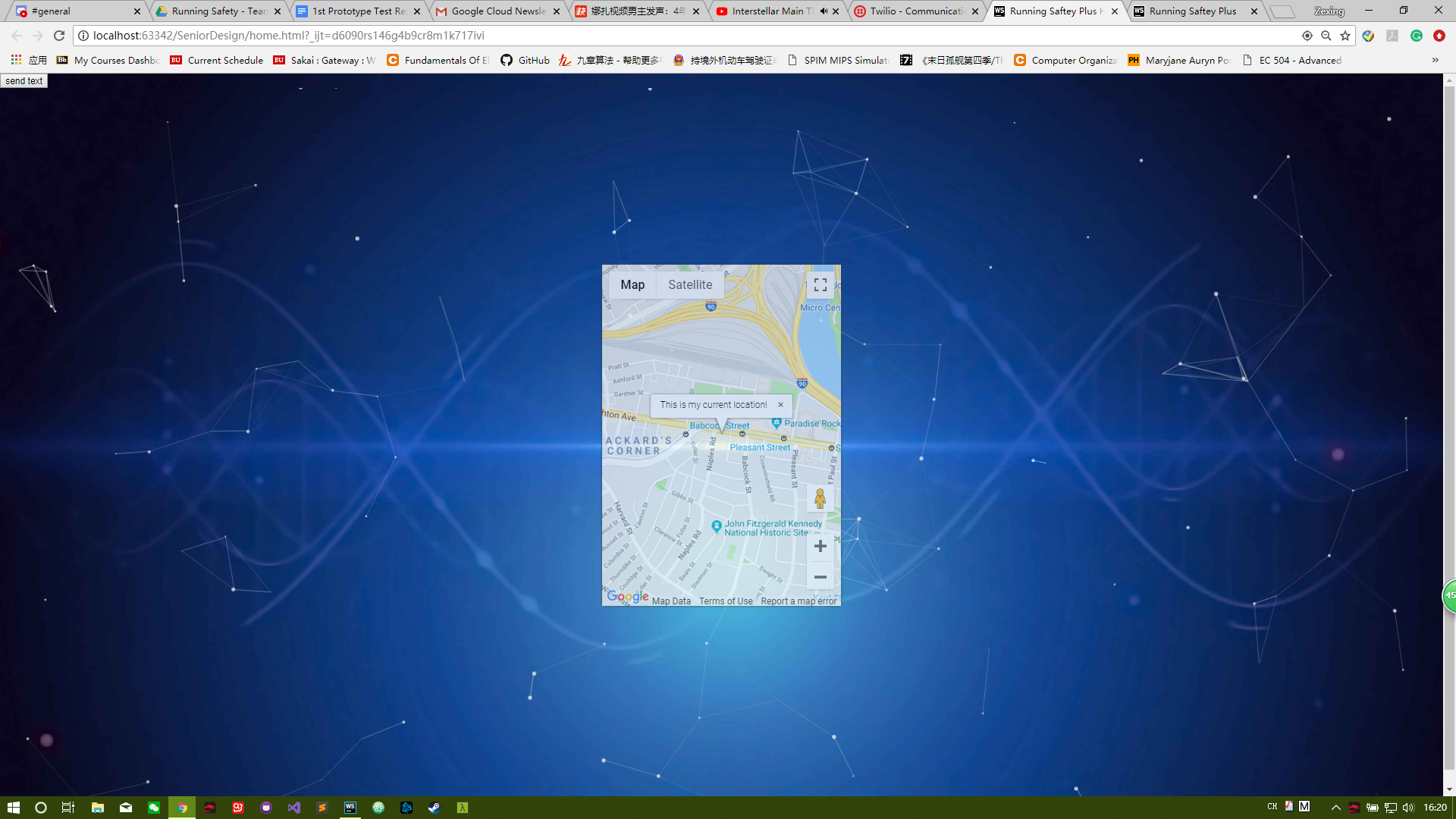
Firebase was used entirely for the authentication process with our web app. We also will be using firebase for database storage to keep it organized and simple. Users can log in easily or choose to signup with they use our web app. Users are given individual userID’s which we will use in the database storage when we need to access certain user’s emergency contacts. All authentication information is stored in the Firebase system that administers can make modification on those accounts.





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 the most powerful API. It can help to solve all of our requirements.

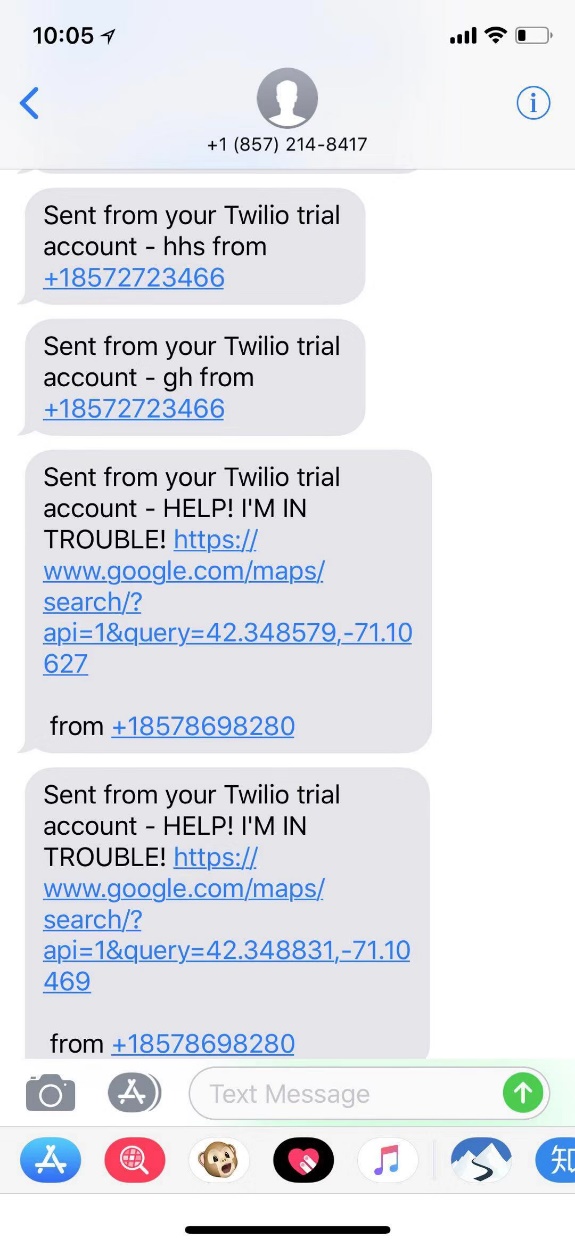


1. *Showed send message function.*

We used Twilio as our platform to send our message. Twilio is a strong platform that can receive & send a text message by calling its API in our application.

*5. Respond message via Twilio*

We can respond message to the user’s phone when the hardware part sends a message to Twilio number. There is a function called Twiml, and then we created a function that can respond to the same message when any text message coming. We can also get the phone number, message, and location used for plotting map on HTML. At last, the user can receive a text message with a link that we can open it on google map. 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.



Measurements of success

1. We successfully opened the webpage and finished the sign-up process by using the information which the professor random given.
2. The given authentication information was given by the professor updated in our firebase system, which shows that the authentication information we just signed up has been updated into the cloud.
3. After the SMS message sent from our hardware device, our Twilio virtual phone number received that information, forward it to the user’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.

Suggestions from the professor:

1. Make the web app as simple and user-friendly as possible since it is a safety device for people of all ages.
2. Implement a “check communication” function before the starting of running or starting the pairing process.

Improvements Needed (Software):

1. User Interface CSS Styling (Make it look better).
2. Implement the database to store data (Emergency Contacts, etc.).
3. Add a new button name “add emergency contacts”. We have to do it after we set up the database.

Conclusion:

In short, we achieved all of our planning goals. Our test is very success expect the battery issue at the very beginning. We definitely are going to improve the circuit and buy some better batteries to make sure the same issue will not happen in the customer installation.