CE-4020-004

Team Omicron

Milestone 2 Cycle Report

Submitted March 31th, 2019

# Executive Summary

LogiSteps is a full stack application that is designed to collect, process, and display user fitness data in a seamless, self-powered construct. LogiSteps allows a user to pair their Bluetooth enabled smart sole with their mobile device and stream data to the cloud in a manner that is unobtrusive and relies very little on the user. Prior to beginning spring quarter, four milestones were set, with the previous being completed late due to miniaturization of design. The completion of this second milestone, and hence, second development cycle, symbolizes the functionality of receiving data over a Bluetooth Low Energy (BLE) data link in the mobile app. This milestone represents the completion of a major hurdle that was preventing full stack data collection.

# Introduction

The Logisteps project has completed it’s fourth week of development in the final quarter of development. The major goals of this quarter are completion of system integration, which includes physical construct to sensors, sensors to microcontroller (both data and power), microcontroller to Android application, Android application to web server, and web server to user. Logisteps initially began the quarter on track, but fell slightly behind in week 3, and has since then adjusted and recovered lost time that occurred in week 3. The main focus for the project during this development cycle (week 3 – week 4) was integration of Bluetooth communication between the microcontroller and the Android application – a task that brought about many complications.

The remainder of this document will go into further detail regarding risks to the project, as well as possible contingency plans that are ready to be implemented if needed, overall performance of the team regarding project plan, the deliverables created for this fifth development cycle, updates to development and planning, followed by a final conclusion. The completion of this second milestone marks a significant pivotal point in Logisteps progress, as it makes full stack data transmission possible.

# Discussion

The completion of this development cycle caps off a major deliverable required for usability of the Logisteps system – the relay of data from a user’s insole to their mobile device using the BLE protocol. While this is not the only progress that was made during this cycle, it is the only major deliverable that was planned for completion. During the completion of this second milestone, other progress has been made in other sub-components of the project, which will be discussed briefly as well.

## Bluetooth Communication and Android App

### Updates to Android App

During the definition of system design, it was decided that data would be communicated from a user’s insole to a mobile application, where that data would then be communicated to a remote server using the mobile application’s existing network stack technology. Several communication protocols were discussed, but the group decided that Bluetooth Low Energy made the most sense for the Logisteps application. Logisteps needed a communication protocol that was extremely low power, simple, wireless, and limited range to a few meters.

During the winter quarter, the mobile application was developed, and Bluetooth communication was established as a proof-of-concept between the mobile device and a Window’s computer. While this proved that Bluetooth connection was possible between the two endpoints, Logisteps’s Bluetooth technology varied significantly from the Window’s PC used for testing purposes. The purpose of this milestone was to establish a Bluetooth connection between the Android application and the mobile application, and then receive the data in the Android application. This would then be demoed to by displaying live data on the Android app.

Prior to beginning work on Bluetooth communication, the Android app was refactored to provide additional stability, and to restructure the infrastructure of the app so that it was more modular and easy to understand, develop, and test. This work began during break week after winter quarter, and continued well into the spring trimester, which partly explains why the reception of Bluetooth communication in the Android app was scheduled for the end of week 4 rather than an earlier week. The app now uses a model, view model, model infrastructure (MVVM), with a repository pattern for data persistence and data retrieval. The figure shown below shows and updated design document for the app’s MVVM design.

A screenshot of a cell phone

Description automatically generated

Figure 1 - Android App's MVVM block diagram.

With this structure, the views simply present data, and make UI changes based on observed objects in their view models and react to user input. All actions that a user create are passed to their respective view models, and the view models perform all logic needed to update data models or query for additional data. The view models communicate with a single model class and use the models to save and retrieve data. In this construct, each layer has a distinct and narrow responsibility, helping reduce error, increasing organization, increasing scalability of the app, and increasing the feasibility of testing.

Data models are only responsible for saving/retrieving data and representing the object and should not be responsible for the implementations in which data is retrieved/saved and should be agnostic to the methods and mediums in which data is stored/retrieved. To satisfy this, the repository architecture that Android recommends is used.



Figure 2 - Android's recommended repository pattern.

Using this pattern, when a view model requests updated data or wishes to save data, it uses a repository interface, which decides whether to retrieve the object from a local database, local cache, or a webservice. This pattern was implemented to improve performance, reduce data usage on metered networks, and decouple data services from the viewmodel and model layers, which increases the testability of the application.

### Bluetooth Communication

Once the Android app was redesigned according to the MVVM and repository pattern as described above, work began to establish Bluetooth connections between the application running on the microcontroller and the application running on the mobile phone. BLE was a new technology for the group, and it involved a lot of learning prior to its implementation. The application developed for the microcontroller was implemented using the Nordic BLE API, and Nordic provides a custom Bluetooth manager for android applications, so the Nordic Android library was used for development.

#### Gatt Server

Bluetooth Low Energy uses a GATT server interface to standardize Bluetooth data communication between two endpoints, and the details of the GATT server are exchanged between the Bluetooth server and Bluetooth client during a handshake on connection. Prior to implementing Bluetooth communication between Logisteps devices, the following GATT service and characteristics was defined.

A screenshot of a cell phone

Description automatically generated

Figure 3 - Logisteps BLE GATT server.

The service represents a logical entity and contains specific data pieces called characteristics. The step service shown in figure 3 has two characteristics, a top sensor characteristic and a bottom sensor characteristic. These characteristics hold pressure readings read off the microcontroller ADC, which are used to represent a step. During the initiation of the Bluetooth connection between Logisteps devices, the Android app must know the UUIDs of the service and characteristics, which were defined as:

* Step service: 00000000-1212-efde-1523-785fef13d123
* Top sensor char: 00001111-1212-efde-1523-785fef13d123
* Bottom sensor char: 00002222-1212-efde-1523-785fef13d123

#### Device Discovery

After defining the GATT server, device discovery was implemented. Upon logging into the app or registering a new account, a user is presented with the main activity, which presents a summary of the user’s step summary. At the top of the UI, as shown in figure 4, a user is able to pair both of their smartsoles with the app. To begin, a user selects “connect” on one of the shoes, which kicks off a BLE device discovery service.

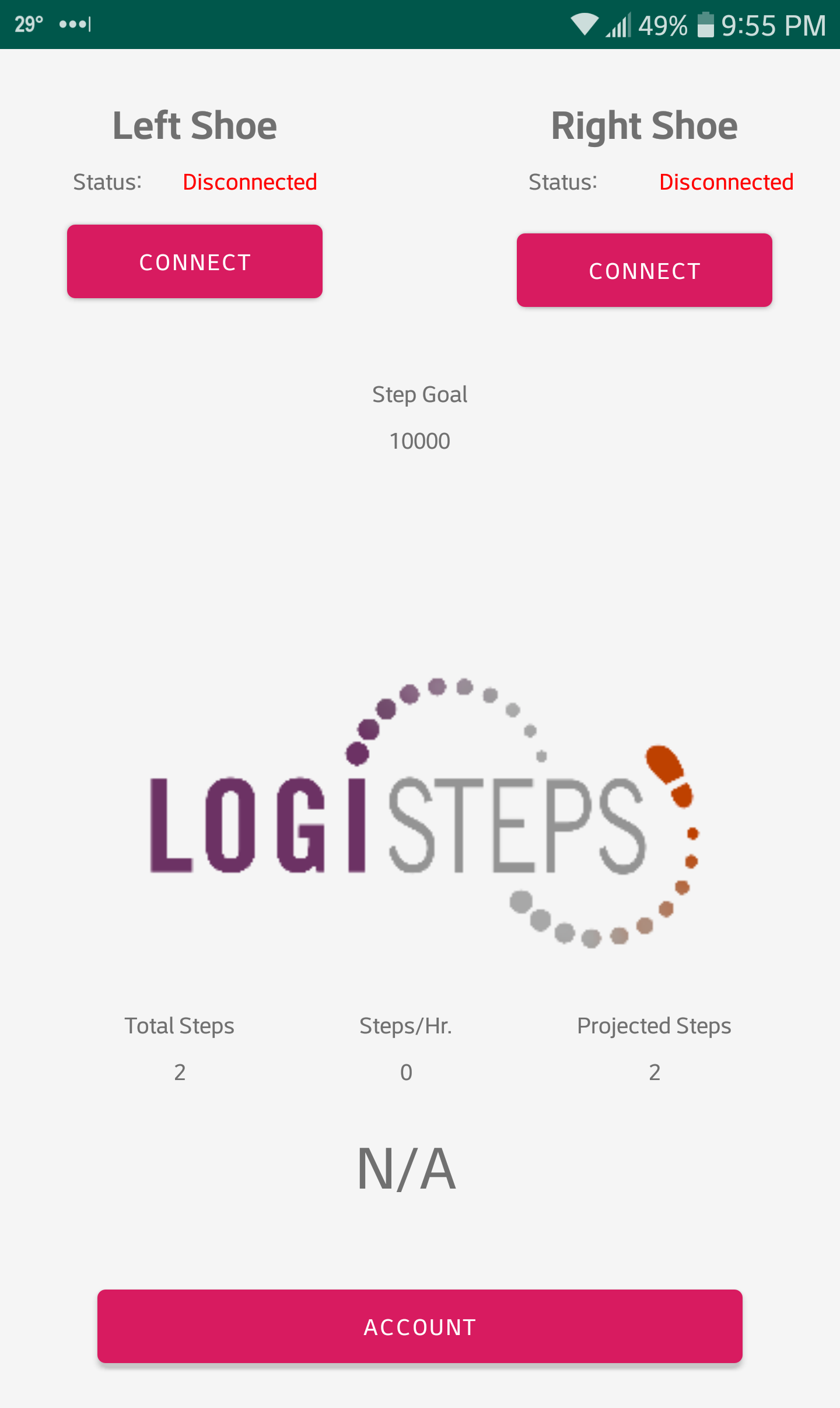


Figure 4 - Android app prior to Bluetooth discovery.

When the discovery service begins, the step summary container is set to invisible, and instead, a list of BLE devices, filtered by the Logisteps service UUID, is displayed in the middle of the screen. As BLE devices are discovered, they are presented to the user using the device name and its MAC address. This function uses the scanner view model, which was implemented by following a Nordic BLE tutorial app. After a user finds their device, they may begin a connection attempt to the device by clicking on the device in the list.

#### Device Connection

Once a user clicks on their device, a connection is established between the device and the Android app. As part of this process, a progress bar is displayed to the user, and a BLE handshake occurs between the mobile app and the microcontroller. A lot of the details of this handshake are handled by the Nordic Bluetooth manager, but during this connection establishment, details of the GATT server are collected and compared against GATT details defined in the Logisteps Bluetooth Manager in the Android app. Once a connection can be established between the devices, callback functions are registered for both GATT characteristics, in which data is parsed and decoded. Once a connection has finished and is deemed successful, the UI is changed using a callback function that hides the progress bar and device list, shows the step summary again, sets the status text for the shoe from “disconnected” to “connected”, and changes the button’s text from “connect” to “disconnect”. At this point, the Android application is ready to receive data.

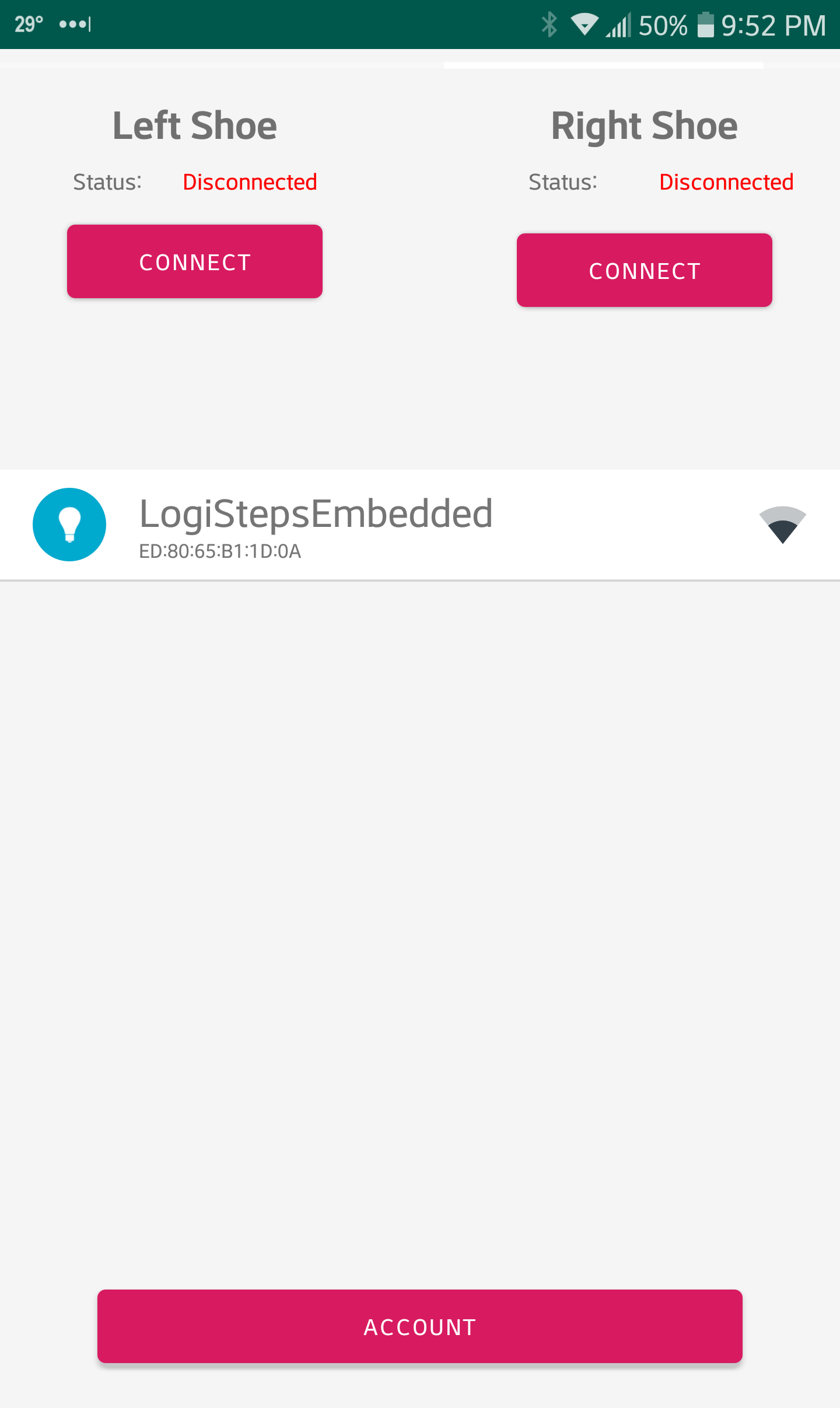


Figure 5 - Android app during device discovery.

#### Data Handling

Once the connection to the BLE device has been established, the device waits for notifications from the microcontroller, which are essentially an event where the microcontroller pushes data to the Bluetooth client. When such an event occurs, a Bluetooth broadcast message is sent from the Android OS, which is handled by the Logisteps app. When the event is received in the app, it calls the callback function previously registered during connection establishment, where the data packet is parsed and decoded. In the callback, the size of the data is checked, and if the data payload does not match the expected size, an error handler is called, which logs a message for debugging and support purposes. If the data is not corrupted, it is converted into an integer, and used to instantiate a step model instance, which is added to a list of steps. Once an appropriate number of steps have been received, the app can then make a POST request to the web server to place the steps into long-term storage.

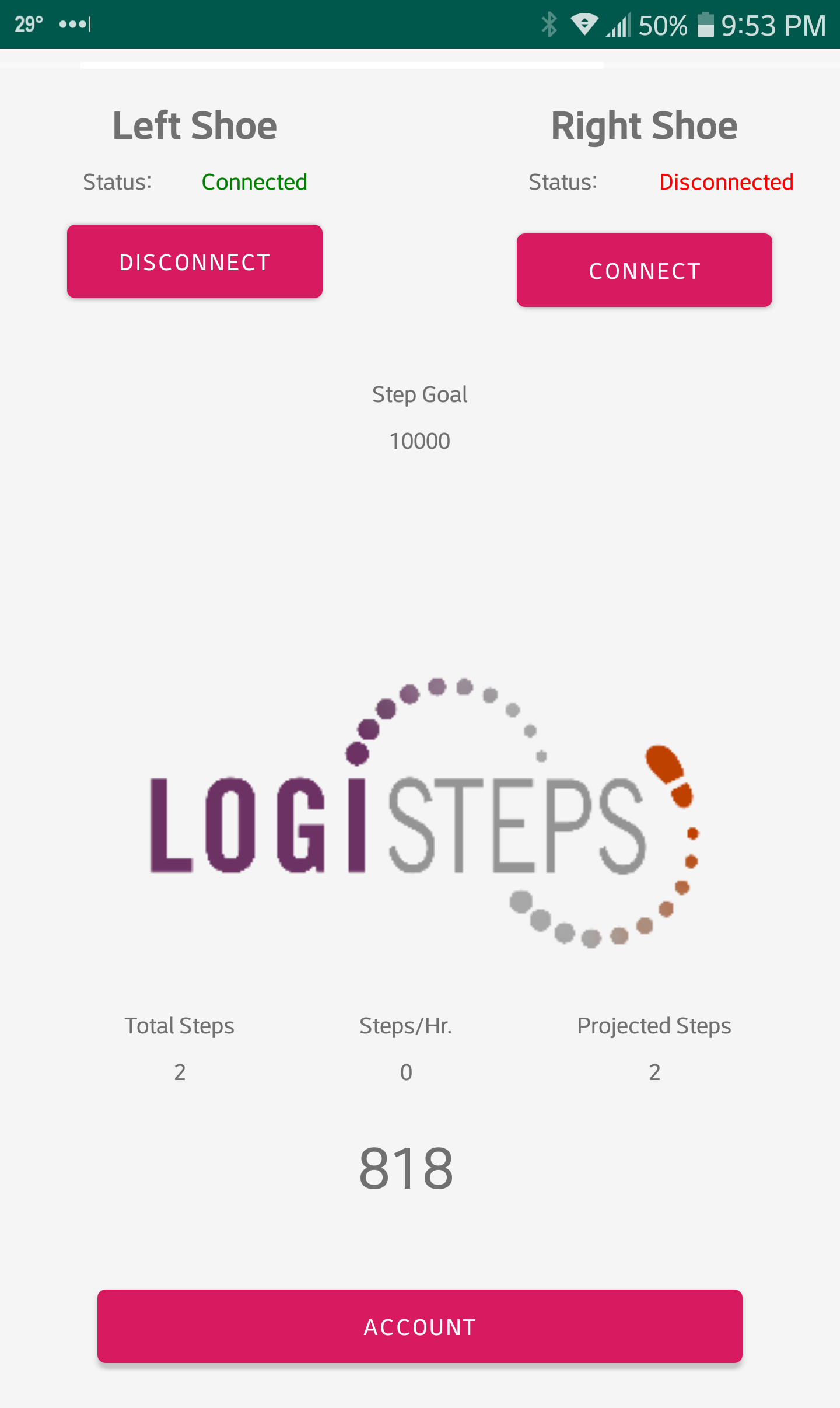


Figure 6 - Bluetooth data being displayed live in UI.

For demo purposes, the live data was displayed on the UI

In future development, additional callbacks will need to be implemented to handle a wide range of BLE related events that can occur, such as link loss, server disconnection, etc. After implementing the functionality described above, the Android application is now at a point where it can connect to Logisteps devices using a custom BLE GATT service, and handle the reception of notifications send from the BLE server over the BLE protocol.

# Risks to the project

The absence of Bluetooth communication between the microcontroller and Android app was the largest risk to the project prior to the completion of this milestone. Without it, it would not have been possible to relay data collected from the piezoelectric sensors to the web server, where the data is ultimately consumed by its end user. By completing this milestone, a huge risk to the project has been mitigated. While the risk to the project has now been greatly reduced, the completion of this milestone does not completely reduce all risk the project faces. In particular, the project still faces the following risks which still need to be addressed in the near future.

1. Backup hardware parts

The Logisteps application currently only has enough hardware parts to satisfy the current needs of the project. The project is not adequately prepared for a situation where the insole needs to be rebuilt – such as the case of damage. To mitigate this risk, additional parts should be purchased immediately.

1. Full Stack Data Transmission

While the ability to receive data greatly reduces the risk of not completing full stack data transmission, it is still a minor risk. Further integration work should be adequate to fully reduce this risk.

1. Mitchell’s Surgery

Mitchell is having surgery on Monday (4/1) and will be out for two days at least.

# Deliverables

The following list summarizes the deliverables completed for milestone two. This is a summarized list of all deliverables and not an exhaustive list of every single task completed.

* Mobile Application re-design
  + Three files were separated into separate modules responsible for specific tasks.
    - Activities
    - Bluetooth
    - Contexts
    - Databases
    - Models
    - Repositories
    - Services
    - ViewModels
  + Increased stability of the app
  + Caching of network resources
* BLE connection and data transmission – implemented in the Android app
* In addition, a test plan has been developed to test functionality of the BLE connection implementation.

In total, several 1000’s of lines of code were written for milestone 2, through dozens of commits to github.

# Development Plan / Plan Update

Implementation of the mobile app was greatly changed during the initial weeks of the spring trimester, and updated documentation has been presented earlier in this document. The redesign of the mobile app fixes several bugs experiences in the initial version of the app and provides a scalable and testable architecture.

Other than the changes in the mobile application, no major development plan changes are needed. In the previous week, the group fell behind in the development plan, but actions were taken to correct his, and the project is now on its initial timeline.

Planned deliverables for the next development cycle are completion of full stack communication, which would complete an initial attempt at full integration. After this milestone has been reached, the remaining work for the project is ironing out bugs and making enhancements to both user experience and data integrity. Additionally, the web application must be brought online using Google Cloud virtual machines to host the database and web server.

# Conclusion

The completion of the second milestone, and second development cycle, brings about a huge reduction in risk to the project. By completing Bluetooth communication between the microcontroller and app, full system integration can now be achieved. The following weeks should bring about full stack communication and completion of the project.